ENRICHED LAWVERE THEORIES
FOR OPERATIONAL SEMANTICS

John C. Baez

Department of Mathematics
University of California
Riverside CA, USA 92521
and
Centre for Quantum Technologies
National University of Singapore
Singapore 117543

Christian Williams

Department of Mathematics
University of California
Riverside CA, USA 92521

email: baez@math.ucr.edu, cwill041@math.ucr.edu

January 18, 2019

Abstract. Enriched Lawvere theories are a generalization of Lawvere theories that allow there to be not merely a *set* of operations of each given arity, but a *graph*, or an object of some other category. Enriched theories can be used to equip systems with operational semantics, and maps between enriching categories can serve to translate between different forms of operational and denotational semantics. We use a definition of Lucyshyn-Wright which allows for theories to be parameterized by a monoidal subcategory of arities, and show that presentation of enriched Lawvere theories is simplified by restricting to natural number arities. We illustrate these ideas with the SKI combinator calculus, a variable-free version of the lambda calculus, presented as a graph-enriched theory.

1. Introduction

Formal systems are sometimes defined without intrinsic connection to how they actually operate in practice. For example, Lawvere theories [13] are an excellent formalism for describing algebraic structures obeying equational laws, but they do not specify how to compute in such a structure, for example taking a complex expression and simplifying it using rewrite rules. Recall that a Lawvere theory is a category with finite products T generated by a single object t, for "type", and morphisms $t^n \to t$ representing n-ary operations, with commutative diagrams specifying equations. There is a Lawvere theory for groups, a Lawvere theory for rings, and so on. We can specify algebraic structures of a given kind in some category C with finite products by a power-preserving functor $\mu\colon \mathsf{T} \to \mathsf{C}$. This is a simple and elegant form of *denotational* semantics. However, Lawvere theories know nothing of *operational* semantics. Our goal here is to address this using "enriched" Lawvere theories [**?**].

In a Lawvere theory the objects are types and the morphisms are terms; the problem is that there are no rewrites between terms, only equations. In operational semantics, program behavior is often specified by labelled transition systems, or labelled directed graphs [21]. The edges of such a graph can be seen as rewrites:

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

To bring rewrites into Lawvere theories we need a structure with types, terms, and also rewrites between terms. This suggests using an enhanced Lawvere theory where instead of merely a *set* of morphisms between objects one has a *graph* or perhaps a *category*. Enriched Lawvere theories are perfectly suited for this purpose.

Using enriched Lawvere theories for operational semantics has been explored in the past. For example, category-enriched theories have been studied by Seely [24], and poset-enriched ones by Ghani and Lüth [16]. Here we allow quite general enrichments, to incorporate these approaches in a common framework. 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 $*$

However, there are many other useful enriching categories. For any enriching category V, a **V-theory** is a V-enriched Lawvere theory with natural number arities (see §4). Even better, there are functors between these which allow the seamless translation between different kinds of operational and denotational semantics. There is a *spectrum* of enriching categories which allow us to examine the semantics of term calculi at various levels of detail:

- **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
  — composition generates morphisms representing *big-step* rewrites.
- **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:
  — a hom-set element represents an *equivalence class* of the symmetric closure of the big-step relation.

Here we take Gph to be the category of reflexive graphs, Set^R, where R is the category with two objects v and e, two morphisms $s, t\colon e \to v$, and a morphism $i\colon v \to e$ obeying $si = ti = 1_v$. Thus, every vertex has a distinguished self-loop, which is needed for the "free category" functor to be a valid change-of-semantics (§6). We could also handle labelled transition systems by a simple augmentation of this theory, but for simplicity we do not consider these.

In section §2, we review Lawvere theories as a more explicit, but equivalent, presentation of finitary monads. In §3, we recall the basics of enrichment, and especially the theory of powers. In §4 we give the central definition of V-theory, from Lucyshyn-Wright [15], which allows us to parametrize our theory by a monoidal subcategory of arities.

In §5 we discuss how functors between enriching categories induce change-of-base 2-functors between their 2-categories of enriched categories, and in §6 we show that functors preserving finite products induce *change-of-semantics*: that is, they map theories to theories and models to models. Our main examples arise from this chain of adjunctions:

$$\begin{tikzcd} \mathsf{Gph} \arrow[r, bend left, "\mathrm{FC}"] \arrow[r, phantom, "\perp"] & \mathsf{Cat} \arrow[l, bend left, "\mathrm{UG}"] \arrow[r, bend left, "\mathrm{FP}"] \arrow[r, phantom, "\perp"] & \mathsf{Pos} \arrow[l, bend left, "\mathrm{UC}"] \arrow[r, bend left, "\mathrm{FS}"] \arrow[r, phantom, "\perp"] & \mathsf{Set} \arrow[l, bend left, "\mathrm{UP}"] \end{tikzcd}$$

The right adjoints here automatically preserve finite products, but the left adjoints do as well, and these are more important in applications:

- Change of base along FC: $\mathsf{Gph} \to \mathsf{Cat}$ maps small-step operational semantics to big-step operational semantics.
- Change of base along FP: $\mathsf{Cat} \to \mathsf{Pos}$ maps big-step operational semantics to full-step operational semantics.
- Change of base along FS: $\mathsf{Pos} \to \mathsf{Set}$ maps full-step operational semantics to denotational semantics.

In §7 we show that models of all possible theories with all possible enrichments can be assimilated into one category using the Grothendieck construction. Finally, in §8 we bring all the strands together by demonstrating these concepts with the SKI-combinator calculus, introducing the idea of developing actual programming languages with enriched Lawvere theories.

Acknowledgements. This paper builds upon the ideas of Mike Stay and Greg Meredith presented in "Representing operational semantics with enriched Lawvere theories" [27]. We appreciate their offer to let us develop this work further for use in the innovative distributed computing system RChain, and gratefully acknowledge the support of Pyrofex Incorporated.

2. Lawvere Theories

Algebraic structures are traditionally treated as sets equipped with operations obeying equations, but we can generalize such structures to live in any category with finite products. For example, given any category C with finite products, we can define a monoid internal to C to consist of:

$$\begin{aligned} \text{an object} \quad & M \\ \text{an identity element} \quad & e\colon 1 \to M \\ \text{and multiplication} \quad & m\colon M^2 \to M \\ \text{obeying the associative law} \quad & m \circ (m \times M) = m \circ (M \times m) \\ \text{and the right and left unit laws} \quad & m \circ (e \times \mathrm{id}_M) = \mathrm{id}_M = m \circ (\mathrm{id}_M \times e). \end{aligned}$$

Lawvere theories formalize this idea. For example, there is a Lawvere theory $\mathsf{Th}(\mathsf{Mon})$, the category with finite products freely generated by an object t equipped with an identity element $e\colon 1 \to t$ and multiplication $m\colon t^2 \to t$ obeying the associative law and unit laws listed above. This captures the "Platonic idea" of a monoid internal to a category with finite products. A monoid internal to C then corresponds to a functor $\mu\colon \mathsf{T} \to \mathsf{C}$ that preserves finite products.

In more detail, let N be any skeleton of the category of finite sets FinSet. Because N is the free category with finite coproducts on 1, N^{op} is the free category with finite products on 1. A **Lawvere theory** is a category with finite products T equipped with a functor $\tau\colon \mathsf{N}^{\mathrm{op}} \to \mathsf{T}$ that is bijective on objects and preserves finite products. Thus, a Lawvere theory is essentially a category generated by one object $\tau(1) = t$ and n-ary operations $t^n \to t$, as well as the projection and diagonal morphisms of finite products.

For efficiency let us call a functor that preserves finite products **cartesian**. Lawvere theories are the objects of a category Law whose morphisms are cartesian functors $f\colon \mathsf{T} \to \mathsf{T}'$ that obey $f\tau = \tau'$. More generally, for any category with finite products C, a **model** of the Lawvere theory T in C is a cartesian functor $\mu\colon \mathsf{T} \to \mathsf{C}$. The models of T in C are the objects of a category $\mathsf{Mod}(\mathsf{T}, \mathsf{C})$, in which the morphisms are natural transformations.

A theory can thus have models in many different contexts. For example, there is a Lawvere theory $\mathsf{Th}(\mathsf{Mon})$, the theory of monoids, described as above. Ordinary monoids are models of this theory in Set, while topological monoids are models of this theory in Top.

For completeness, it is worthwhile to mention the *presentation* of a Lawvere theory: after all, we are arguing their utility in everyday programming. How exactly does the above "sketch" of $\mathsf{Th}(\mathsf{Mon})$ produce a category with finite products? It is precisely analogous to the presentation of an algebra by generators and relations: we form the free category with finite products on the data given, and impose the required equations. The result is a category whose objects are powers of M, and whose morphisms are composites of products of the morphisms in $\mathsf{Th}(\mathsf{Mon})$, projections, deletions, symmetries and diagonals. A detailed account was given by Barr and Wells [3, Chap. 4]; for a more computer-science-oriented approach see Crole [7, Chap. 3].

Currently, monads are more widely used in computer science than Lawvere theories. However, Hyland and Power have suggested that Lawvere theories could do much of the work that monads do today [8]. In 1965, Linton [14] proved that Lawvere theories correspond to "finitary monads" on the category of sets. For every Lawvere theory T, there is an adjunction:

$$\begin{tikzcd} \mathsf{Set} \arrow[r, bend left, "F"] \arrow[r, phantom, "\perp"] & \mathsf{Mod}(\mathsf{T}, \mathsf{Set}). \arrow[l, bend left, "U"] \end{tikzcd}$$

CAN YOU MAKE THIS LOOK BETTER??? THE ARROWS ARE CLOSE AT LEFT AND FAR APART AT RIGHT, AND THE F, U ARE CLOSER TO THE LEFT. The underlying set functor

$$U\colon \mathsf{Mod}(\mathsf{T}, \mathsf{Set}) \to \mathsf{Set}$$

sends each model μ to the image of the generating object in Set, $X = \mu(\tau(1))$. Its left adjoint, the free model functor

$$F\colon \mathsf{Set} \to \mathsf{Mod}(\mathsf{T}, \mathsf{Set}),$$

sends each finite set n to the representable functor $\mathsf{T}(|n|, -)\colon \mathsf{T} \to \mathsf{Set}$, and in general any set X to the functor that maps $t^n \in \mathsf{T}$ to the set of all n-ary operations on X: $\{f(x_1, ..., x_n) | f \in \mathsf{T}(n, 1), x_i \in X\}$. WHAT DOES THAT MEAN??? These give an adjunction

$$\mathsf{Mod}(F(n), \mu) = \mathsf{Mod}(\mathsf{T}(|n|, -), \mu) \cong \mu(n) \cong \mu(1)^n = \mathsf{Set}(n, U(\mu))$$

where the left isomorphism arises from the Yoneda lemma, and the right isomorphism from the product preservation of μ.

This adjunction induces a monad T on Set:

$$T(X) = \int^{n \in \mathsf{N}} X^n \times \mathsf{T}(n, 1). \tag{1}$$

The integral here is a coend, essentially a coproduct quotiented by the equations of the theory and the equations induced by the cartesian structure of the category. The monad constructed this way is always **finitary**: that is, it preserves filtered colimits [1].

Conversely, for a monad T on Set, its Kleisli category $\mathsf{Kl}(T)$ is the category of all free algebras of the monad, which has all coproducts. There is a functor $k\colon \mathsf{Set} \to \mathsf{Kl}(T)$ that is the identity on objects and preserves coproducts (because it is a right adjoint). Thus,

$$k^{\mathrm{op}}\colon \mathsf{Set}^{\mathrm{op}} \to \mathsf{Kl}(T)^{\mathrm{op}}$$

is a cartesian functor, and restricting its domain to N^{op} is a Lawvere theory. When T is finitary, the monad arising from this Lawvere theory is naturally isomorphic to T itself. For more details see [3, 13, 19].

This correspondence sets up an equivalence between the category Law of Lawvere theories and the category of finitary monads on Set. There is also an equivalence of between the category $\mathsf{Mod(T)}$ of models of any given Lawvere theory and the category of algebra of the corresponding finitary monad T. Furthermore, all this generalizes with Set replaced by any "locally finitely presentable" category [1].

3. Enrichment

To allow more general semantics, we now turn to Lawvere theories that have hom-*objects* rather than mere hom-*sets*. To do this we use enriched category theory [10], where we replace sets with objects of a monoidal category $(\mathsf{V}, \otimes, I)$, the "enriching" category. A **V-category** or V-enriched category C is:

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

such that composition is associative and unital. A **V-functor** $F\colon \mathsf{C} \to \mathsf{D}$ is:

$$\begin{array}{rl} \text{a function} & F_0\colon \mathrm{Ob}(\mathsf{C}) \to \mathrm{Ob}(\mathsf{D}) \\ \text{a collection of morphisms} & F_{ab}\colon \mathsf{C}(a,b) \to \mathsf{D}(F_0(a), F_0(b)) \quad \forall a,b \in \mathsf{C} \end{array}$$

such that F is compatible with composition and identity. A **V-natural transformation** $\alpha\colon F \Rightarrow G$ is:

$$\text{a family} \quad \alpha_a\colon I \to \mathsf{D}(F_0(a), G_0(a)) \quad \forall a \in \mathrm{Ob}(\mathsf{C})$$

such that α is "natural" in a. There a 2-category VCat of V-categories, V-functors, and V-natural transformations.

To go further, let V be a closed symmetric monoidal category, providing

$$\begin{array}{rl} \text{internal hom} & [-,-]\colon \mathsf{V}^{\mathrm{op}} \times \mathsf{V} \to \mathsf{V} \\ \text{tensor-hom adjunction} & \mathsf{V}(u \otimes v, w) \cong \mathsf{V}(u, [v,w]) \quad \forall u,v,w \in \mathrm{Ob}(\mathsf{V}) \\ \text{braiding natural isomorphism} & \tau_{u,v}\colon u \otimes v \cong v \otimes u \quad \forall u,v \in \mathrm{Ob}(\mathsf{V}) \end{array}$$

obeying the hexagon and triangle equations and the symmetry axiom saying $\tau_{u,v}$ is the inverse of $\tau_{v,u}$ [17]. Then we can construct new V-categories from old by taking tensor products and opposites in an obvious way. Moreover V is itself a V-category, denoted $\tilde{\mathsf{V}}$, with the hom-object function providing the internal hom. The tensor-hom adjunction is called "currying" in computer science; the counit is "evaluation".

Generalizing how the closed symmetric monoidal category V has a tensor product and internal hom, a V-category can be "V-tensored" and "V-powered". Suppose C is any V-category. Given

$a \in \mathsf{C}$ and $v \in \mathsf{V}$, we say an object $v \odot a \in \mathsf{C}$ is a **tensor product** of a by v if it is equipped with isomorphisms

$$\mathsf{C}(v \odot a, b) \cong [v, \mathsf{C}(a, b)] \tag{2}$$

V-natural in b. In the special case $\mathsf{V} = \mathsf{Set}$ this forces $v \cdot a$ to be the v-fold coproduct of copies of a:

$$v \odot a = \sum_{i \in v} a.$$

In general, if we can choose a tensor product $v \odot a$ for all $v \in \mathsf{V}$, $a \in C$ we say C is **V-tensored**, and we obtain a V-functor

$$\odot \colon \tilde{\mathsf{V}} \otimes \mathsf{C} \to \mathsf{C}.$$

For example, $\tilde{\mathsf{V}}$ is always V-tensored via the tensor product $\otimes \colon \tilde{\mathsf{V}} \otimes \tilde{\mathsf{V}} \to \tilde{\mathsf{V}}$.

Similarly, given $b \in \mathsf{C}$ and $v \in \mathsf{V}$, we say an object $v \pitchfork b \in \mathsf{C}$ is a **power** of b by v if it is equipped with isomorphisms

$$[v, \mathsf{C}(a, b)] \cong \mathsf{C}(a, v \pitchfork b). \tag{3}$$

V-natural in a. In the special case $\mathsf{V} = \mathsf{Set}$ this forces $v \pitchfork a$ to be the v-fold product of copies of b:

$$v \pitchfork b = \prod_{i \in v} b.$$

In general, if we can choose a power $v \odot b$ for all $v \in \mathsf{V}$, $b \in C$ we say C is **V-powered**, and we obtain a V-functor

$$\pitchfork \colon \tilde{\mathsf{V}}^{\mathrm{op}} \otimes \mathsf{C} \to \mathsf{C}.$$

For example, $\tilde{\mathsf{V}}$ is always V-powered via the internal hom $[-,-] \colon \mathsf{V}^{\mathrm{op}} \times \mathsf{V} \to \mathsf{V}$.

When C is both V-tensored and V-powered, we have isomorphisms

$$\mathsf{C}(v \odot a, b) \cong [v, \mathsf{C}(a, b)] \cong \mathsf{C}(a, v \pitchfork b). \tag{4}$$

that are V-natural in $a, b \in \mathsf{C}$ and $v \in \mathsf{V}$. When $\mathsf{V} = \mathsf{Set}$ this occurs if C has all products and coproducts. In what follows we use exponential notation $b^v := v \pitchfork b$ for V-powers, and denote the unit I by 1, because the enriching categories we consider are cartesian. We should note that tensors are sometimes called "copowers", and powers are sometimes called "cotensors".

There are just a few more technicalities. A category is **locally finitely presentable** if it is the category of models for a finite limits theory, and an object is **finite** if its representable functor is **finitary**: that is, it preserves filtered colimits [1]. A V-category C is **locally finitely presentable** if its underlying category C_0 is locally finitely presentable, C has finite powers, and $(-)^x \colon \mathsf{C}_0 \to \mathsf{C}_0$ is finitary for all finitely presentable x. The details are not crucial here: all categories to be considered are locally finitely presentable. We will use V_f to denote the full subcategory of V of finite objects: in Gph, these are simply graphs with finitely many vertices and edges.

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. These are greatly generalized in the enriched case, because whereas the only finite objects in Set are *finite sets*, there are more complex finite objects in any other enriching V. In the next section we discuss some difficulties of this generality and how to circumvent them.

4. Enriched Lawvere theories

Power introduced the notion of enriched Lawvere theory about twenty years ago, "in seeking a general account of what have been called notions of computation" [22]. The original definition is as follows: for a symmetric monoidal closed category $(\mathsf{V}, \otimes, I)$, a "V-enriched Lawvere theory" is a V-category T that has powers by objects in V_f, equipped with an identity-on-objects V-functor

$$\tau \colon \mathsf{V}_f^{\mathrm{op}} \to \mathsf{T}$$

that preserves these powers. A "model" of a V-theory is a V-functor $\mu \colon \mathsf{T} \to \mathsf{V}$ which preserves powers by objects of V_f. There is a category $\mathsf{Mod}(\mathsf{T}, \mathsf{V})$ whose objects are models and whose morphisms are V-natural transformations. The monadic adjunction and equivalence of §2 generalize to this context.

However, this requires T to have all powers by all objects of V_f, i.e., the theory must have arities for every finite object of V. These *generalized arities* may be very powerful—rather than only inputting n-tuples of terms, we can operate on any finite object of terms! But despite the great potential, this idea has remained essentially dormant for decades, partly for the difficulty of intuition, but also that of presentation: while it is easy to inductively generate all n-ary operations from binary, unary, nullary ones, it is certainly more subtle to generate powers for all finite objects.

Power has also introduced"enriched sketches" as a way of presenting enriched Lawvere theories, [11], and "monads with arities" have been explored by Berger, Melliés and Weber [5], but these do not yet appear to have be widely understood . What does it really mean for an operation to take in a finite graph of terms? To what subjects does this pertain primarily, and how can we learn to use this generality? We hope that someone answers these questions, so that we can use more general arities in computer programming.

For this paper, however, 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 [15], which allows for theories to be parameterized by a **system of arities**, a full monoidal subcategory

$$j \colon \mathsf{J} \hookrightarrow \mathsf{V}.$$

Associated to J there is a V-category we call $\tilde{\mathsf{J}}$ whose objects are those of J and whose hom-objects are given by

$$\hom(a, b) = [j(a), j(b)].$$

Definition 1. Given a system of arities $j \colon \mathsf{J} \hookrightarrow \mathsf{V}$, a V-enriched algebraic theory with j-arities or **J-V theory** (T, τ) is a V-category T equipped with a V-functor

$$\tau \colon \tilde{\mathsf{J}}^{\mathrm{op}} \to \mathsf{T}$$

that is bijective on objects and **J-power preserving**: that is, it preserves powers by objects of (the subcategory image of) J.

While the literature demands that τ be the identity on objects, we use a weaker definition to handle change-of-base.

Definition 2. A **model** of T in a V-category C is a J-power preserving V-functor

$$\mu \colon \mathsf{T} \to \mathsf{C}.$$

In the same way that all the objects of a Lawvere theory are N-powers of a generating object, the objects of a J-V theory are J-powers t^J of a generating object $t = \tau(I)$, one for each $J \in \mathsf{J}$. (BUT IN THE LAST SECTION WE SAID WE'D BE USING 1 AS OUR NAME FOR THE MONOIDAL

UNIT!) Note that t itself is t^I. Just as every $n \in \mathsf{N}^{\mathrm{op}}$ is a power of 1, every $J \in \tilde{\mathsf{J}}^{\mathrm{op}}$ is a power of the monoidal unit $I \in \mathsf{V}$, i.e. using Eq. (4) for $\tilde{\mathsf{J}}^{\mathrm{op}}$, $(\star \mapsto 1_J) \in [I, \tilde{\mathsf{J}}^{\mathrm{op}}(J,J)]$ is sent to the canonical isomorphism:

$$J \cong I^J. \tag{5}$$

This is just the opposite of the usual isomorphism $J \cong J^I$. Then, since τ preserves J-powers, this implies that every object of T is a power of $t = \tau(I)$.

There is a category VLaw whose objects are J-V theories and whose morphisms are J-power-preserving V-functors $f\colon \mathsf{T} \to \mathsf{T}'$ such that $f\tau = \tau'$. For every J-V theory T and every V-category C with J-powers, we define the category of models $\mathsf{Mod}(\mathsf{T},\mathsf{C})$ to consist of J-power preserving V-functors $\mathsf{T} \to \mathsf{C}$ and V-natural transformations. If V is a **cosmos**, i.e. complete and cocomplete, then the functor categories of VCat are also V-categories, including VLaw and $\mathsf{Mod}(\mathsf{T},\mathsf{C})$. This is potentially very useful, and the enriching categories $\mathsf{V} = \mathsf{Gph}, \mathsf{Cat}, \mathsf{Pos}, \mathsf{Set}$ that are the focus of this paper are indeed cosmoi.

Here is an overview of the concepts:

$$\begin{array}{ccccl}
j\colon & \mathsf{J} & \hookrightarrow & \mathsf{V} & \text{arities} \\
--- & --- & --- & --- & \text{enrichment} \\
\tau\colon & \tilde{\mathsf{J}}^{\mathrm{op}} & \to & \mathsf{T} & \text{theory} \\
 & & & \downarrow & \text{model} \\
 & & & \mathsf{C} & \text{semantic } \mathsf{V}\text{-category}
\end{array}$$

This parameterization is quite general; for example, Power's definition is the case $\mathsf{J} = \mathsf{V}_f$. A system of arities is **eleutheric** if left Kan extensions along j exist and are preserved by $\mathsf{V}(K,-)$ for all $K \in Ob(\mathsf{J})$. This is what is needed to have the essential monadicity theorems: Lucyshyn-Wright proved that any J-V theory for an eleutheric system of arities has a category of models for $\mathsf{C} = \mathsf{V}$ which is monadic over V.

The usual kinds of arities are eleutheric: in particular, finite cardinals. We take these as the objects of our system of arities J. WHEN??? STARTING RIGHT NOW???

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

$$n_{\mathsf{V}} := n \cdot I_{\mathsf{V}} = \sum_{i \in n} I_{\mathsf{V}}$$

characterized by the universal property

$$\mathsf{V}(n_{\mathsf{V}}, a) = \mathsf{V}(I_{\mathsf{V}} \odot n, a) \cong \mathsf{Set}(n, \mathsf{V}(I_{\mathsf{V}}, a)). \tag{6}$$

For $\mathsf{J} = \mathsf{N}_{\mathsf{V}}$, we will call J-V theories **V-theories** for simplicity.

Because N_{V} is eleutheric, V-theories correspond to V-monads on V, just as ordinary Lawvere theories correspond to monads on Set, and §6 will demonstrate that the arities are essentially the same—but now we have the rich "operational" information of V, and this V is adaptable.

How exactly does the "free-forgetful" V-adjunction work?

$$\begin{tikzcd} \mathsf{V} \arrow[r, bend left, "F"] \arrow[r, phantom, "\perp"] & \mathsf{Mod}(\mathsf{T},\mathsf{V}) \arrow[l, bend left, "U"] \end{tikzcd}$$

Lucyshyn-Wright [15, Sec. 8] has described the **free model** of a V-theory. This is the enriched generalization of the free model described in §2: it is the composite of the (opposite) theory

$\tau^{\mathrm{op}}\colon \mathsf{N_V} \to \mathsf{T}^{\mathrm{op}}$ with the Yoneda embedding $y\colon \mathsf{T}^{\mathrm{op}} \to [\mathsf{T}, \mathsf{V}]$, sending each $n_{\mathsf{V}} \in \mathsf{N_V}$ to its representable V-functor, i.e. the n-ary operations of T:

$$\begin{array}{ccccc} \mathsf{N}_{\mathsf{V}}^{\mathrm{op}} & \xrightarrow{\tau} & \mathsf{T}^{\mathrm{op}} & \xrightarrow{y} & [\mathsf{T}, \mathsf{V}] \\ n_{\mathsf{V}} & \mapsto & t^{n_{\mathsf{V}}} & \mapsto & \mathsf{T}(t^{n_{\mathsf{V}}}, -) \end{array}$$

Since an object of V does not necessarily have a "poset of finite subobjects" over which to take a filtered colimit (as in Set), the extension of the free model to all of V is specified by a somewhat higher-powered generalization: the "free model" functor on V is the *left Kan extension* of $y\tau$ along j.

$$\begin{tikzcd} \mathsf{N_V} \arrow[rr, "y\tau"] \arrow[dr, "j"'] & \arrow[d, phantom, "\Downarrow\eta"] & {[\mathsf{T}, \mathsf{V}]} \\ & \mathsf{V} \arrow[ur, dotted, "F:=\mathrm{Lan}_j y\tau"'] & \end{tikzcd}$$

This is essentially the universal "best solution" to this diagram commuting; i.e. for any other functor $G\colon \mathsf{V} \to [\mathsf{T}, \mathsf{V}]$ and V-natural transformation $y\tau \Rightarrow Gj$, the latter factors uniquely through the unit η.

The forgetful adjoint $U\colon [\mathsf{T}, \mathsf{V}] \to \mathsf{V}$ is still evaluation at the generating object, and hence the V-monad has a more concrete (elementwise) formula as an enriched coend:

$$T(V) = \int^{n_{\mathsf{V}} \in \mathsf{N_V}} V^{n_{\mathsf{V}}} \times \mathsf{T}(t^{n_{\mathsf{V}}}, t) \tag{7}$$

Example 3. Let $\mathsf{V} = \mathsf{Cat}$ and let $\mathsf{T} = \mathsf{Th}(\mathrm{PsMon})$ be the Cat-theory of pseudomonoids [12]. Now rather than associativity and unitality *equations*, there are 2-isomorphisms called the **associator** and **unitors** which rewrite in the direction of a "normal form". To equate the multiple possible rewrite sequences, we need 2-dimensional coherence conditions. Here is the presentation of the 2-theory:

Th(PsMon)

sorts	M	pseudomonoid
operations	$e\colon 1 \to M$	identity
	$m\colon M^2 \to M$	multiplication
rewrites	$\alpha_{MMM}\colon m \circ (m \times \mathrm{id}_M) \Rightarrow m \circ (\mathrm{id}_M \times m)$	associator
	$\lambda\colon m \circ (e \times \mathrm{id}_M) \Rightarrow \mathrm{id}_M$	left unitor
	$\rho\colon m \circ (\mathrm{id}_M \times e) \Rightarrow \mathrm{id}_M$	right unitor
coherence laws	$(M \times \alpha) \circ \alpha_{MmM} \circ (\alpha \times M) = \alpha_{MMm} \circ \alpha_{mMM}$	pentagon identity
	$(M \times \lambda) \circ \alpha_{M1M} = \rho \times M$	triangle identity

NOT OBVIOUS FROM HERE WHERE α_{MmM} and α_{M1M} come from!!!

Models of T in Cat are monoidal categories, and the induced 2-monad on Cat is the "free monoidal category" 2-monad. Let us explore this example in more detail: a model of $\mathsf{T} = \mathrm{FC}_{fp}(\mathsf{Th}(\mathrm{PsMon}))$

is a finite-product-preserving 2-functor $\mu\colon \mathsf{T} \to \mathsf{Cat}$, which sends

$$\begin{array}{rclcl} t & \mapsto & \mathsf{C} & & \\ m & \mapsto & \otimes\colon & \mathsf{C}^2 \to \mathsf{C} & \\ e & \mapsto & I\colon & 1 \to \mathsf{C} & \\ \alpha & \mapsto & \otimes \circ (\otimes \times 1_{\mathsf{C}}) & \Rightarrow & \otimes \circ (1_{\mathsf{C}} \times \otimes) \\ \lambda & \mapsto & I \circ 1_{\mathsf{C}} & \Rightarrow & 1_{\mathsf{C}} \\ \rho & \mapsto & 1_{\mathsf{C}} \circ I & \Rightarrow & 1_{\mathsf{C}} \end{array}$$

such that the coherence laws of the rewrites are preserved. Thus, a model is a category equipped with a tensor product $\otimes$ and unit object I such that these operations are unital and associative up to natural isomorphism; so these models are precisely monoidal categories. In this way, 2-theories generalize equipping *set*-like objects with algebraic structure to *category*-like objects.

To form the free model on a category $\mathsf{C} \in \mathsf{Cat}$, we follow the above method: the formula for left Kan extension (writing n instead of n_{Cat} for simplicity, see §6) gives $F(\mathsf{C})\colon \mathsf{T} \to \mathsf{V}$ by

$$F(\mathsf{C}) = \int^{n \in \mathsf{N}_{\mathsf{Cat}}} \mathsf{T}(t^n, t^{(-)}) \times \mathsf{C}^n$$

which is constructed by pairing n-ary morphisms in T with n-tuples of objects in C for all $n \in \mathsf{N}_{\mathsf{Cat}}$, then quotienting the coproduct of these pairs by the equations of T and C.

This functor is not very intuitive; but composing with the left adjoint, i.e. evaluating $F(\mathsf{C})$ at 1, gives the *free monoidal category* on C: in the same way that the (underlying set of the) free monoid on a set X consists of all finite strings of elements of X, $F(\mathsf{C})(1)$ consists of all finite tensors of objects and morphisms of C, and all composites of these morphisms, up to the relations induced by the (composites and tensors of the) images of the associator and unitors.

In general for each $m \in \mathsf{N}_{\mathsf{Cat}}$, $F(\mathsf{C})(m)$ gives the category of all m-tuples of elements of $F(C)(1)$, forming the "free monoidal category on C with m variables", like a polynomial ring. This can be useful for imposing further relations on $F(\mathsf{C})$, analogous to Galois theory. Though pure category theorists does not usually consider free variables, perhaps this is a useful notion even outside the context of computation.

The free monoidal category can be given a computational presentation as in the judgement tables for $\mathrm{FC}_{fp}(\mathsf{Th}(\mathsf{Mon}))$ in §2, with the universal property of the product replaced with the bilinearity of the tensor product. There is surely a systematic method of generating these presentations, but it is probably still implicit in the literature for sketches.

Finally, an algebra of the monad

$$T := F(-)(1)\colon \mathsf{Cat} \to \mathsf{Cat} :: \mathsf{C} \mapsto \int^n \mathsf{C}^n \times \mathsf{T}(n, 1)$$

is a category A equipped with a functor $\otimes_A\colon F(A)(1) \to A$ such that it is compatible with the multiplication and unit of the monad, which are the "free" tensor bifunctor and monoidal unit on A. Hence, $(A, \otimes_A)$ is precisely a monoidal category, and we have the equivalence:

$$\mathrm{PsMon}(\mathsf{Cat}) = \mathsf{Mod}(\mathsf{T}, \mathsf{Cat}) \simeq \mathrm{Alg}(T).$$

5. Change of Base

We now have the tools to formulate the main idea: certain V correspond to certain kinds of *semantics*, and changing enrichments corresponds to a *change of semantics*. We propose a general framework in which one can translate between different forms of semantics: small-step, big-step, full-step operational semantics, and denotational semantics.

This translation is effected by a (strong) **monoidal functor**: a functor

$$(F, \lambda, \upsilon)\colon (\mathsf{V}, \otimes_{\mathsf{V}}, I_{\mathsf{V}}) \to (\mathsf{W}, \otimes_{\mathsf{W}}, I_{\mathsf{W}})$$

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

$$\begin{aligned} \lambda\colon &\quad F(a) \otimes_{\mathsf{W}} F(b) \cong F(a \otimes_{\mathsf{V}} b) \\ \upsilon\colon &\quad I_{\mathsf{W}} \cong F(I_{\mathsf{V}}) \end{aligned}$$

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

This induces a **change of base** functor $F_*\colon \mathsf{VCat} \to \mathsf{WCat}$ [6]. 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. The W-category $F_*(\mathsf{C})$ is defined as follows:

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

If $f\colon \mathsf{C} \to \mathsf{D} \in \mathsf{VCat}$ is a V-functor, then $F_*(f)_{\mathrm{obj}} = f_{\mathrm{obj}}$ and $F_*(f)_{\mathrm{hom}} = F \circ f_{\mathrm{hom}}$. If $\alpha\colon f \Rightarrow g$ is a V-natural transformation and $c \in \mathsf{C}$, then $F_*(\alpha)_c := F(\alpha_c) \circ \upsilon$.

Hence, the change of base operation forms a 2-functor (or "Cat-functor"):

$$\begin{array}{ccc} \mathsf{MonCat} & \xrightarrow{(-)_*} & \mathsf{2Cat} \\ (F\colon \mathsf{V} \to \mathsf{W}) & \mapsto & (F_*\colon \mathsf{VCat} \to \mathsf{WCat}) \end{array}$$

In particular, there is an important correspondence of adjunctions (if V has all coproducts of I_{V}):

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

Each set X is represented in V as the X-indexed coproduct of the unit object, and conversely each object v of V is represented in Set by the hom-set from the unit to v. The latter induces the "underlying (Set-)category" change of base, which forgets the enrichment. The former induces the "free V-enrichment" change of base, whereby ordinary Set-categories are converted to V-categories, denoted $\mathsf{C} \mapsto \tilde{\mathsf{C}}$. These form an adjunction, because 2-functors preserve adjunctions.

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

We only need to show that everything is simplified by restricting to this particular J.

6. Simplifying with natural number 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. The idea is that instead of thinking about fancy enriched powers, we are justified in considering ordinary products.

Let V be a cartesian closed category with finite coproducts of the unit object, and let N_V be defined as above.

Lemma 4. The functors $[n_\mathsf{V}, -] \colon \mathsf{V} \to \mathsf{V}$ and $(-)^n \colon \mathsf{V} \to \mathsf{V}$ are naturally isomorphic, i.e. n_V-powers in $\tilde{\mathsf{V}}$ are isomorphic to n-powers (n-fold products) in V.

Proof. If $a, b \in \mathsf{V}$, then

$$\begin{array}{rcll} \mathsf{V}(a, [n_\mathsf{V}, b]) & \cong & \mathsf{V}(a \times n_\mathsf{V}, b) & \text{hom-tensor adjunction} \\ & = & \mathsf{V}(a \times (n \cdot I_\mathsf{V}), b) & \text{definition of } n_\mathsf{V} \\ & \cong & \mathsf{V}(n \cdot (a \times I_\mathsf{V}), b) & \text{distributivity} \\ & \cong & \mathsf{V}(n \cdot a, b) & \text{unitality} \\ & \cong & \mathsf{V}(a, b)^n & \text{cocontinuity of hom} \\ & \cong & V(a, b^n) & \text{continuity of hom.} \end{array}$$

Each of these isomorphisms is natural in a and b; hence by the Yoneda lemma, $[n_\mathsf{V}, -] \cong (-)^n$. $\square$

So, the full sub-V-category $\tilde{\mathsf{N}}_\mathsf{V}$ has hom-objects which behave like the exponentiation of N:

$$[n_\mathsf{V}, m_\mathsf{V}] \cong (m \cdot I_\mathsf{V})^n \cong (m^n)_\mathsf{V}.$$

In VCat, the objects of the theory T are n_V-powers of a generating object s. Alas, we cannot simply say that " $s^{n_\mathsf{V}} \cong s^n$ ", because the latter does not type-check in the V-category T: products are characterized by a Set-enriched universal property. However, we only need:

Lemma 5. Let T be a V-category with N_V-powers, and let $s \in \mathsf{T}$. Then a hom into s^{n_V} is isomorphic to n homs into s:

$$\mathsf{T}(a, s^{n_\mathsf{V}}) \cong [n_\mathsf{V}, \mathsf{T}(a, s)] \cong \mathsf{T}(a, s)^n$$

by definition of power, and Lemma 1.

We want to know when the functor $F \colon \mathsf{V} \to \mathsf{W}$ induces a change of base $F_* \colon \mathsf{VCat} \to \mathsf{WCat}$ which "preserves enriched-theories"—if by F every V-theory τ_V gives rise to a W-theory τ_W, then F is a *change of semantics*. That is, given a V-theory

$$\tau_\mathsf{V} \colon \tilde{\mathsf{N}}_\mathsf{V}^{\mathrm{op}} \to \mathsf{T}$$

we want to determine a minimal condition for the base-changed functor

$$F_*(\tau_\mathsf{V}) \colon F_*(\tilde{\mathsf{N}}_\mathsf{V}^{\mathrm{op}}) \to F_*(\mathsf{T})$$

to induce a W-theory in a canonical way. So, assuming there is a clear identification of $F_*(\tilde{\mathsf{N}}_\mathsf{V}^{\mathrm{op}})$ and $\tilde{\mathsf{N}}_\mathsf{W}^{\mathrm{op}}$, it suffices to require that $F_*(\tau_\mathsf{V})$ preserves N_W-powers.

Because $F_*(-)_{\mathrm{hom}}$ is defined

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

combined with the previous lemmas, the preservation of "$\mathsf{N}_{(-)}$-power preserving functors" by F_* is implied by the preservation of finite products by F: and since our enriching categories and base-change functors are cartesian, this is automatic.

Lemma 6. Let $F \colon \mathsf{V} \to \mathsf{W}$ be a cartesian functor, and let N_V, N_W be defined as above. If $f \colon \mathsf{C} \to \mathsf{D}$ is a V-functor which preserves N_V-powers, then $F_*(f) \colon F_*(\mathsf{C}) \to F_*(\mathsf{D})$ is a W-functor which preserves N_W-powers.

Proof.

$$\begin{array}{rcll} F_*(\mathsf{D})(F_*(f)(a), F_*(f)(s^{n_\mathsf{V}})) & = & F(\mathsf{D}(f(a), f(s^{n_\mathsf{V}})) & \text{definition of base change} \\ & \cong & F(\mathsf{D}(f(a), f(s)^{n_\mathsf{V}}) & f \text{ preserves } \mathsf{N_V}\text{-powers} \\ & \cong & F(\mathsf{D}(f(a), f(s))^n) & \text{Lemma 2 for } \mathsf{V} \\ & \cong & F(\mathsf{D}(f(a), f(s)))^n & F \text{ cartesian} \\ & = & F_*(\mathsf{D})(f(a), f(s))^n & \text{definition of base change} \\ & \cong & F_*(\mathsf{D})(f(a), f(s)^{n_\mathsf{W}}) & \text{Lemma 2 for } \mathsf{W} \end{array}$$

□

Finally, let $\tilde{n}\colon \tilde{\mathsf{N}}_\mathsf{W} \to F_*(\tilde{\mathsf{N}}_\mathsf{V})$ be the isomorphism which sends $n_\mathsf{W} \mapsto n_\mathsf{V}$ and is the identity on morphisms. We can then construct a W-functor which precisely fits the definition of a W-theory:

Theorem 7. Let V, W be cartesian closed categories with finite coproducts of their unit objects, and let $F\colon \mathsf{V} \to \mathsf{W}$ be a cartesian functor. Then F is a **change of semantics**; i.e. for every V-theory $\tau_\mathsf{V}\colon \tilde{\mathsf{N}}_\mathsf{V}^{\mathrm{op}} \to \mathsf{T}$, the W-functor

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

is a W-theory. Moreover, F preserves *models*, i.e. for every $\mathsf{N_V}$-power preserving V-functor $\mu\colon \mathsf{T} \to \mathsf{C}$, the W-functor $F_*(\mu)$ preserves $\mathsf{N_W}$-powers.

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

Hence, any cartesian functor between cartesian closed categories constitutes a "change of semantics" — this is a simple, ubiquitous condition, which provides for a method of translating formal languages between various "modes of operation".

Moreover, this reasoning generalizes to **multisorted** V-theories, enriched theories which have multiple sorts: given any $n \in \mathbb{N}$, the monoidal subcategory $(\mathsf{N_V})^n$ is also an eleutheric system of arities. At the end of §8, we give an example demonstrating why this is a very useful generalization.

Before exploring applications, we introduce two more useful kinds of translations, and demonstrate how all of this information be encapsulated in one categorical notion.

7. The category of all V-theories

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

$$f^*\colon \mathsf{VMod}(\mathsf{T}', \mathsf{C}) \to \mathsf{VMod}(\mathsf{T}, \mathsf{C})$$

defined as precomposition with f. Similarly, given a cartesian functor $g\colon \mathsf{C} \to \mathsf{C}'$, this induces a "change-of-model" functor

$$g_*\colon \mathsf{VMod}(\mathsf{T}, \mathsf{C}) \to \mathsf{VMod}(\mathsf{T}, \mathsf{C}')$$

defined as postcomposition with g.

These translations, as well as change-of-base, can all be packed up nicely using the **Grothendieck construction**: given a (pseudo)functor $F\colon \mathsf{D} \to \mathsf{Cat}$, there is a category $\int F$ that encapsulates all of the categories in the image of F, defined as follows:

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

Moreover there is a functor $\overline{F}\colon \int F \to \mathsf{D}$ given as follows:

$$\begin{array}{rl} \text{on objects} & \overline{F}\colon (d,x) \mapsto d \\ \text{on morphisms} & \overline{F}\colon (f,a) \mapsto f. \end{array}$$

For more details see [6, 9]. We noted in §4 that VLaw and $\mathsf{Mod}(\mathsf{T},\mathsf{C})$ are V-categories when V is a cosmos: this and other conditions imply we can use the *enriched* Grothendieck construction [4]; but we will focus on the Set-enriched case for simplicity.

This idea allows us to bring together all of the different enrichments, theories, and models into one big category. For every enriching category V, let VCat_{np} be the subcategory of VCat of V-categories with N-powers and N-power preserving functors; then there is a functor

$$\mathsf{VMod}\colon \mathsf{VLaw}^{\mathrm{op}} \times \mathsf{VCat}_{np} \to \mathsf{Cat}$$

which sends (T,C) to $\mathsf{VMod}(\mathsf{T},\mathsf{C})$. The (bi)functoriality of VMod gives the contravariant change-of-theory and the covariant change-of-model above.

Using the Grothedieck construction, we obtain a category $\int \mathsf{VMod}$, with a morphism

$$((f,g),\alpha)\colon ((\mathsf{T},\mathsf{C}),\mu) \to ((\mathsf{T}',\mathsf{C}'),\mu')$$

being finite power-preserving V-functors $f\colon \mathsf{T} \to \mathsf{T}'$, $g\colon \mathsf{C} \to \mathsf{C}'$, and V-natural transformation $\alpha\colon \mathsf{VMod}(f,g)(\mu) \to \mu'$.

Lemma 8. There is a functor

$$\mathrm{thy}\colon \mathsf{CCC} \to \mathsf{Cat}$$

which assigns V to $\int \mathsf{VMod}$ and $(F\colon \mathsf{V} \to \mathsf{W})$ to a functor $(F_*^*\colon \int \mathsf{VMod} \to \int \mathsf{WMod})$.

Proof. Given $F\colon \mathsf{V} \to \mathsf{W}$, base change $F_*\colon \mathsf{VCat} \to \mathsf{WCat}$ is a 2-functor, thereby inducing the functor $F_*^*\colon \mathsf{VMod} \to \mathsf{WMod}$ 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 Grothendieck construction once more to encapsulate even the enrichment:

Theorem 9. There is a category $\mathrm{Thy} := \int \mathrm{thy}$ with a morphism

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

being a cartesian functor F and a morphism $(f,g,\alpha)\colon F_*^*(((\mathsf{T},\mathsf{C}),\mu)) \to ((\mathsf{T}',\mathsf{C}'),\mu')$ in WMod.

This category assimilates a whole lot of useful information. Most importantly, there are morphisms between objects of "different kinds", something we consider often but is normally not possible in category theory. For example, in Thy there is a morphism:

$$(\mathsf{Set},((u_{\mathsf{Grp}},\mathrm{Disc}),\exp))\colon (\mathsf{Set},((\mathsf{T}_{\mathrm{Grp}},\mathsf{Set}),(\mathbb{R},+,0))) \to (\mathsf{Set},((\mathsf{T}_{\mathsf{Grp}},\mathsf{Top}),(\mathbb{R},\times,1)))$$

There are many unexplored questions about the large, heterogeneous categories which arise from the Grothendieck construction, regarding what unusual structure may be gained, such as limits and colimits with objects of different types, or identifying "processes" in which the kinds of objects change in an essential way. This is just a remark; for our purposes we need only recognize that enriched Lawvere theories can be assimilated into one category, which provides a unified context for change-of-base, change-of-theory, and change-of-modelling.

8. Applications

In theoretical computer science literature, enriched algebraic theories have primarily been studied in the context of "computational effects". Mike Stay and Greg Meredith have recognized that Lawvere theories can actually be utilized for the design of *programming languages* [28]. This idea comes from an important but underappreciated subject in foundations — 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*:

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

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

Despite its apparent simplicity, there are 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 variable 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** [23], 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. A great introduction into the world of strange and powerful combinators can be found in [26].

The key insight of Stay and Meredith [27] is that Lawvere theories are by definition free of variables, and it is precisely through abstraction elimination that a programming language can be made an algebraic object. When representing a computational calculus as an 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-calculus:

Th(SKI)

sort	t	
term constructors	$S\colon$	$1 \to t$
	$K\colon$	$1 \to t$
	$I\colon$	$1 \to t$
	$(-\ -)\colon$	$t^2 \to t$
structural congruence	n/a	
rewrites	$\sigma\colon$	$(((S\ -)\ \ =)\ \ \equiv) \Rightarrow ((-\ \ \equiv)\ (=\equiv))$
	$\kappa\colon$	$((K\ -)\ \ =) \Rightarrow -$
	$\iota\colon$	$(I\ -) \Rightarrow -$

These rewrites are implicitly universally quantified; i.e. they apply to arbitrary subterms $-, =, \equiv$ without any variable binding involved, by using the cartesian structure of the category. They are simply edges with vertices:

$$\begin{tikzcd}
(((S\ -)\ \ =)\ \ \equiv)\colon \arrow[d, Rightarrow, "\sigma"'] & t^3 \arrow[r, "l^{-1}\times t^3"] \arrow[d, equal] & 1\times t^3 \arrow[r, "S\times t^3"] & t^4 \arrow[r, "(-\ -)\times t^2"] \arrow[d, phantom, "\Downarrow"] & t^3 \arrow[r, "(-\ -)\times t"] & t^2 \arrow[r, "(-\ -)"] & t \arrow[d, equal] \\
((-\ \ \equiv)\ (=\equiv))\colon & t^3 \arrow[r, "t^2\times\Delta"] & t^4 \arrow[r, "t\times\tau\times t"] & t^4 \arrow[rr, "(-\ -)\times(-\ -)"] & & t^2 \arrow[r, "(-\ -)"] & t
\end{tikzcd}$$

$$\begin{tikzcd}
((K\ -)\ \ =)\colon \arrow[d, Rightarrow, "\kappa"'] & t^2 \arrow[r, "l^{-1}\times t^2"] \arrow[d, equal] & 1\times t^2 \arrow[r, "K\times t^2"] & t^3 \arrow[r, "(-\ -)\times t"] \arrow[d, phantom, "\Downarrow"] & t^2 \arrow[r, "(-\ -)"] & t \arrow[d, equal] \\
-\colon & t^2 \arrow[rr, "t\times !"] & & t\times 1 \arrow[rr, "r"] & & t
\end{tikzcd}$$

$$\begin{tikzcd}
(I\ -)\colon \arrow[d, Rightarrow, "\iota"'] & t \arrow[r, "l^{-1}"] \arrow[d, equal] & 1\times t \arrow[r, "I\times t"] \arrow[dr, phantom, "\Downarrow"] & t^2 \arrow[r, "(-\ -)"] & t \arrow[d, equal] \\
-\colon & t \arrow[rrr, "t"] & & & t
\end{tikzcd}$$

These abstract rules are evaluated on concrete terms by "plugging in" via precomposition:

$$\begin{tikzcd}
((KS)I)\colon \arrow[d, Rightarrow, "\kappa\circ(S\times I)"'] & 1 \arrow[r, "S\times I"] \arrow[d, equal] & t^2 \arrow[r, "((K\ -)\ =)"] \arrow[d, phantom, "\Downarrow"] & t \arrow[d, equal] \\
S\colon & 1 \arrow[r, "S\times I"] & t^2 \arrow[r, "-"] & t
\end{tikzcd}$$

(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.)

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

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

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 $\mu(t)^2 \cong \mu(t^2)$, a vertex $(u\ v)$

in $\mu(t)$ which is their application. In this way we get all possible terms (writing $\mu(S), \mu(K), \mu(I)$ as S, K, I for simplicity):

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

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,

$$\mu_{1,t}\colon \mathsf{Th}(SKI)(1,t) \to \mathsf{Gph}(1,\mu(t))$$

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 $\mu(a) \Rightarrow \mu(b)$.

These rewrites in the image of μ are *graph transformations*, 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 $a \Rightarrow b\colon 1 \to \mathsf{C}$ is just a morphism $a(1) \to b(1)$ in C, a rewrite of closed terms $a \Rightarrow b\colon 1 \to \mu(t)$ 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:

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

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 $\mathsf{Th}(SKI)$, 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 adjunction of §4:

$$\begin{tikzcd}
\mathsf{Gph} \arrow[r, bend left, "f_{\mathsf{Gph}}"] \arrow[r, phantom, "\perp"] & \mathsf{Mod}(\mathsf{Th}(SKI), \mathsf{Gph}) \arrow[l, bend left, "u_{\mathsf{Gph}}"]
\end{tikzcd}$$

Then the canonical model of closed terms and rewrites is simply the free model on the empty graph, $f_{\mathsf{Gph}}(\emptyset)$, i.e. the V-functor $\mathsf{T}(1,-)\colon \mathsf{T} \to \mathsf{V}$. Hence for us, the syntax and semantics of the SKI combinator calculus are unified in the model

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

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

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

Interestingly, in the free model on a nonempty graph, the vertices represent designated "ground variables", and edges represent rewrites of one variable into another. This is potentially useful for "building in" a language with other basic features not intrinsic to the theory.

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 FC: $\mathsf{Gph} \to \mathsf{Cat}$ 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 is cartesian, 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:

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

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

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

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

The same reasoning applies to the "free poset" functor FP: $\mathsf{Cat} \to \mathsf{Pos}$; 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. Since there was no real algebraic information in the free category, this is simply adding the property that all the distinct paths between two terms are equal, while retaining transitivity.

Finally, we can pass to the purely abstract realm where all computation is already complete. One rarely speaks of the "free set on a poset", but there is a left adjoint FS: $\mathsf{Pos} \to \mathsf{Set}$ of the functor UP: $\mathsf{Set} \to \mathsf{Pos}$ sending any set to the discrete poset on that set. The functor FP collapses every connected component of the poset to a point; equating every formal expression to its final value (this is cartesian because . 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.

So, from this simple sequence of functors, we can translate between the main kinds of semantics for the SKI-calculus. For example, we have the following computation:

$$\begin{tikzcd}
 & (((S\ K)\ (I\ K))\ S) \arrow[dl, "\iota"'] \arrow[dr, "\sigma"] \arrow[ddl, dotted, "\iota\sigma"] \arrow[ddl, dotted, bend left, "\sigma\iota"] \arrow[ddr, dotted, "\kappa\sigma"] \arrow[ddr, dotted, bend right, "\kappa\sigma\iota"] \arrow[ddr, dotted, bend right=40, "\kappa\iota\sigma"] & \\
(((S\ K)\ K)\ S) \arrow[d, "\sigma"'] & & ((K\ S)\ ((I\ K)\ S)) \arrow[d, "\kappa"] \arrow[dll, "\iota"'] \\
((K\ S)\ (K\ S)) \arrow[rr, "\kappa"'] & & S
\end{tikzcd}$$

The solid arrows are the one-step rewrites of the initial Gph-theory; applying FC_* gives the dotted composites, and FP_* asserts that all composites between any two objects are equal. Finally, FS_* collapses the whole diagram to S. This is a simple demonstration of the basic stages of computation: small-step, big-step, full-step, and denotational semantics.

Of course, most interesting languages are not always terminating, confluent, nor deterministic; the "spectrum" of semantics being presented here is simply an initial proof-of-concept. We expect that there are more interesting change-of-base functors which handle these subtleties — they have likely been studied in other contexts.

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 $\mathsf{Th}(SKI)$ by adding a reduction context marker as a unary operation, and a structural congruence rule which pushes the marker to the left-hand side of an application; lastly we modify the rewrite rules to be valid only when the marker is present:

Th(SKI+R)

sort	t	
term constructors	$S, K, I\colon$	$1 \to t$
	$R\colon$	$t \to t$
	$(-\ -)\colon$	$t^2 \to t$
structural congruence	$R(x\ y) = (Rx\ y)$	
rewrites	$\sigma_r\colon$	$(((RS\ -)\ =)\ \equiv) \Rightarrow ((R-\ \equiv)\ (=\ \equiv))$
	$\kappa_r\colon$	$((RK\ -)\ =) \Rightarrow R-$
	$\iota_r\colon$	$(RI\ -) \Rightarrow R-$

The SKI-calculus is thereby equipped with "lazy evaluation", an essential paradigm in modern programming. This represents a broad potential application of equipping theories with computational methods, such as evaluation strategies.

Moreover, these equipments can be added or removed as needed: using change-of-theory, we can utilize a "free reduction" Gph-functor $f_R\colon \mathsf{Th}(SKI) \to \mathsf{Th}(SKI+R)$:

objects	t^n	$\mapsto$	t^n
hom-vertices	S, K, I	$\mapsto$	S, K, I
	$(-\ -)$	$\mapsto$	$R(-\ -)$
hom-edges	σ, κ, ι	$\mapsto$	$\sigma_r, \kappa_r, \iota_r$

This essentially interprets ordinary SKI as having every subterm be a reduction context. This is a Gph-functor because its hom component consists of graph-homomorphisms:

$$f_{n,m} \colon \mathsf{Th}(SKI)(t^n, t^m) \to \mathsf{Th}(SKI + R)(t^n, t^m)$$

which simply send each application to its postcomposition with R, and each rewrite to its "marked" correspondent; and this is all coherent: for example, even though $(((S\ x)\ y)\ z) \mapsto R(R(R(S\ x)\ y)\ z)$, the extra markers are ignored by σ_r, because they are now just a part of the lefthand terms.

So, by precomposition this induces the change of theory on categories of models:

$$f_R^* \colon \mathsf{Mod}(\mathsf{Th}(SKI + R), \mathsf{C}) \to \mathsf{Mod}(\mathsf{Th}(SKI), \mathsf{C})$$

for all semantic categories C, which forgets the reduction contexts.

Similarly, there is a Gph-functor $u_R \colon \mathsf{Th}(SKI{+}R) \to \mathsf{Th}(SKI)$ which forgets reduction contexts, by sending $\sigma_r, \kappa_r, \iota_r \mapsto \sigma, \kappa, \iota$ and $R \mapsto id_t$; this latter is the only way that the marked reductions can be mapped coherently to the unmarked. However, this means that u_R^* does not give the desired change-of-theory of "freely adjoining contexts", because collapsing R to the identity eliminates the significance of the marker.

This illustrates a key aspect of categorical universal algebra: because change-of-theory is given by precomposition and is thus contravariant, *properties* (equations) and *structure* (operations) can only be removed.

This is a necessary limitation, at least in the present setup, but there are ways of working around it: of course, these abstract theories are not floating in isolation but are implemented in code. One can simply use a "maximal theory" with all pertinent structure, then selectively forget as needed.

8.4. **Multisorted: the ρ-calculus.** Many algebraic theories involve multiple sorts in an essential way. In concurrency theory, *process calculi* exhibit an ontology which is fundamentally distinct from that of sequential computing — rather than simply expressing a series of terms and rewrites, these calculi represent dynamical systems of communicating processes.

The π-calculus, designed by Milner [20], consists of **names** and **processes**, or *channels* and *agents* which communicate on those channels. Far more than a sequence of instructions on a single machine, computation develops through the interaction of independent participants in a network.

This powerful idea of modern computer science is being utilized by Greg Meredith and Mike Stay to design a deeply cooperative distributive computing system, called RChain. The "R" stands for "reflective higher-order π-calculus", or **ρ-calculus**. It is like Milner's original language, with one crucial difference: "reflection" is a formal system's ability to turn code into data and vice versa. This is a powerful idea which replaces opaque, atomic variables with transparent, anatomical names, or "quoted processes" [18].

Utilizing both reflection and combinators in a theory requires special type discipline; there is a designated auxiliary sort T for combinatory terms which are analogous to "machine code", as contrasted with the sorts N and P which are to be thought of as the actual language: see [27] §7.3 for the details of the translation. The presentation below is only a fragment; it has yet to be determined how to best represent the full algebraic theory of the ρ-calculus; we expect that a true mathematical characterization of reflection will call for original and enlightening ideas.

Th(RHO)

Sorts	N		names	T		terms	
	P		processes				
Operations	$0\colon$	$1 \to P$	null process	$S\colon$	$1 \to T$	combinator	
	$\&\colon$	$P \to N$	code to data	$K\colon$	$1 \to T$	combinator	
	$*\colon$	$N \to P$	data to code	$(-\ -)\colon$	$T^2 \to T$	application	
	$!\colon$	$N \times P \to P$	send				
	$?\colon$	$N^2 \times P \to P$	receive				
	$- \mid -\colon$	$P^2 \to P$	parallel				
Equations		$(P,\ \mid,\ 0)$	commutative monoid				
Rewrites	$\gamma\colon$	$x?(y).P \mid x!(z) \mid Q \Rightarrow$	$P[z/y] \mid Q$	$\sigma\colon$	$(((S\ -)\ \ =)\ \ \equiv) \Rightarrow$	$((-\ \ \equiv)\ (= \equiv))$	
	$\epsilon\colon$	$*(\&(P)) \Rightarrow$	P	$\kappa\colon$	$((K\ -)\ \ =) \Rightarrow$	$-$	

9. Conclusion

We have established the basics of how enriched Lawvere theories provide a framework for unifying the syntax and semantics, the structure and behavior of formal languages. Enriching theories in category-like structures reifies operational semantics by incorporating rewrites between terms; and cartesian functors between enriching categories induce change-of-semantics functors between categories of models—this simplified condition is obtained by using only finite cardinal arities.

This base-change, along with change-of-theory and change-of-modelling, can be assimilated into one category using an iterated Grothendieck construction Thy, which consists of all enriched Lawvere theories. Finally, enriched theories can be used not only for computational effects but the actual design of concrete programming languages, through the use of combinators.

References

[1] J. Adámek and J. Rosický, *Locally Presentable and Accessible Categories*, Cambridge U. Press, Cambridge, 1994. (Referred to on page 4, 5, 6.)

[2] J. Baez and M. Stay, Physics, topology, logic and computation: a Rosetta Stone, in *New Structures for Physics*, ed. B. Coecke, Springer, Berlin, 2011, pp. 95–172. Available as arXiv:0903.0340. (Referred to on page 15.)

[3] M. Barr and C. Wells, *Toposes, Triples and Theories*, reprinted in *Repr. Theory Appl. Categ.* **12** (2005). Available at http://www.tac.mta.ca/tac/reprints/articles/12/tr12abs.html. (Referred to on page 4, 5.)

[4] J. Beardsley and L. Z. Wong, The enriched Grothendieck construction. Available as arXiv:1804.03829. (Referred to on page 14.)

[5] C. Berger, P.-A. Melliès and M. Weber, Monads with arities and their associated theories, *J. Pure Appl. Algebra*, **216** (2012), 2029–2048. Available as arXiv:1101.3064. (Referred to on page 7.)

[6] F. Borceux, *Handbook of Categorical Algebra*, vol. 2, Cambridge U. Press, Cambridge, 1994. (Referred to on page 11, 14.)

[7] R. Crole, *Categories for Types*, Cambridge U. Press, Cambridge, 1993. (Referred to on page 4.)

[8] M. Hyland and J. Power, The category theoretic understanding of universal algebra: Lawvere theories and monads, in *Electron. Notes Theor. Comput. Sci.* **172** (2007), 437–458. (Referred to on page 4.)

[9] B. Jacobs, *Categorical Logic and Type Theory*, Elsevier, Amderstam, 1999. (Referred to on page 14.)

[10] G. M. Kelly, *Basic Concepts of Enriched Category Theory*, reprinted in *Repr. Theory Appl. Categ.* **10** (2005), 1–136. Available at http://www.tac.mta.ca/tac/reprints/articles/10/tr10abs.html. (Referred to on page 5.)

[11] Y. Kinoshita, J. Power and M. Takeyama, Sketches, *J. Pure Appl. Algebra* **143** (1999), 275–291. (Referred to on page 7.)

[12] B. Day and R. Street, Monoidal categories and Hopf algebroids, *Adv. Math.* **127** (1997), 99–157. (Referred to on page 9.)

[13] F. W. Lawvere, Functorial semantics of algebraic theories, reprinted in *Repr. Theory Appl. Categ.* **5** (2004), 1–121. Available at http://tac.mta.ca/tac/reprints/articles/5/tr5abs.html. (Referred to on page 1, 5.)

[14] F. E. J. Linton, Some aspects of equational theories, in *Proceedings of the Conference on Categorical Algebra*, eds. S. Eilenberg et al., Springer, Berlin, 1965. (Referred to on page 4.)

[15] R. B. B. Lucyshyn-Wright, Enriched algebraic theories and monads for a system of arities, *Theory Appl. Categ.* **31** (2016), 101–137. Available at http://www.tac.mta.ca/tac/volumes/31/5/31-05abs.html. (Referred to on page 2, 7, 8.)

[16] C. Lüth and N. Ghani, Monads and modular term rewriting, *Category Theory and Computer Science (Santa Margherita Ligure, 1997)*, Springer, Berlin, 1997, pp. 69–86. Available at http://www.informatik.uni-bremen.de/c̃xl/papers/ctcs97l.pdf. (Referred to on page 2.)

[17] S. Mac Lane, *Categories for the Working Mathematician*, Springer, Berlin, 1998. (Referred to on page 5.)

[18] L. G. Meredith and M. Radestock, A reflective higher-order calculus, *Electronic Notes in Theoretical Computer Science* **141** (2005), 49–67. (Referred to on page 20.)

[19] B. Milewski, *Category Theory for Programmers*, Chap. 14: Lawvere theories, 2017. Available at https://bartoszmilewski.com/2017/08/26/lawvere-theories/. (Referred to on page 5.)

[20] R. Milner, The polyadic π-calculus: a tutorial, in *Logic and Algebra of Specification*, Springer, Berlin, 1993, 203–246. (Referred to on page 20.)

[21] G. D. Plotkin, A structural approach to operational semantics, *J. Log. Algebr Program.* **60/61** (2004) 17–139. Available at http://homepages.inf.ed.ac.uk/gdp/publications/sos_jlap.pdf. (Referred to on page 2.)

[22] J. Power, Enriched Lawvere theories, *Theory Appl. Categ.* **6**(1999), 83–93. (Referred to on page 7.)

[23] M. Schönfinkel, Über die Bausteine der mathematischen Logik, *Mathematische Annalen* **92** (1924), 305–316. Available at http://www.digizeitschriften.de/dms/img/?PID=GDZPPN002270110. (Referred to on page 15.)

[24] R. A. G. Seely, Modelling computations: a 2-categorical framework, in *Proceedings of the Second Annual IEEE Symposium on Logic in Computer Science (LICS 1987)*, IEEE Computer Society Press, Ithaca, New York, pp. 22–25. (Referred to on page 2.)

[25] P. Selinger, Lecture notes on the lambda calculus. Available as arXiv:0804.3434. (Referred to on page .)

[26] R. Smullyan, *To Mock a Mockingbird: and Other Logic Puzzles Including an Amazing Adventure in Combinatory Logic*, Oxford U. Press, Oxford, 2000. (Referred to on page 15.)

[27] M. Stay and L. G. Meredith, Representing operational semantics with enriched Lawvere theories. Available as arXiv:1704.03080. (Referred to on page 3, 15, 20.)

[28] M. Stay and L. G. Meredith, Logic as a distributive law. Available as arXiv:1610.02247. (Referred to on page 15.)