ENRICHED LAWVERE THEORIES FOR OPERATIONAL SEMANTICS DRAFT VERSION

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

May 3, 2019

Abstract. Enriched Lawvere theories are a generalization of Lawvere theories that allow us to describe the operational semantics of formal systems. For example, a graph-enriched Lawvere theory describes structures that have a *graph* of operations of each arity, where the vertices are operations and the edges are rewrites between operations. 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 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 not always explicitly connected to how they operate in practice. Lawvere theories [18] 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 theory for groups, a theory for rings, and so on. We can specify algebraic structures of a given kind in some category C with finite products by a product-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; however there are no relations between terms, only equations. The process of computing one term into another should be given by hom-objects with more structure. In operational semantics, program behavior is often specified by labelled transition systems, or labelled directed graphs [28]. The edges of such a graph represent rewrites:

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

We can use an enhanced Lawvere theory in which, rather than merely *sets* of morphisms, there are *graphs* or perhaps a *categories*. Enriched Lawvere theories are exactly for this purpose.

To be clear, this is certainly not a new idea. Using enriched Lawvere theories for operational semantics has been explored in the past. For example, category-enriched theories have been studied by Seely [32] for the λ-calculus, and poset-enriched ones by Ghani and Lüth [21] for understanding "modularity" in term rewriting systems. They have been utilized extensively by Power, enriching in ω-complete partial orders to study recursion [29] – in fact, there the simplified "natural number" enriched theories which we explore were implicitly considered.

In the context of these works, the purpose of the present paper is to provide a simple, general exposition of enriched theories: we hope to familiarize computer scientists with enriched category theory, and prove some basic results to show that one does not need to leave the nice computational world of cartesian closed categories to enjoy the benefits of enrichment.

While we consider general enriching categories, we focus on graph-enriched Lawvere theories, which have a clear connection to the original idea of operational semantics:

sort	:	generating object
term constructors	:	generating morphisms
structural congruence	:	commuting diagrams
$*$ rewrite rules	:	generating hom-edges $*$

For an enriching category V, a **V-theory** is a V-enriched Lawvere theory with natural number arities (see §4). There are functors between these which allow the 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:

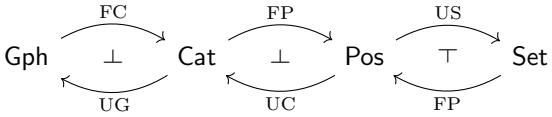
- **Graphs**: Gph-theories represent "small-step" operational semantics:
  — a hom-graph edge is a *single term* rewrite.
- **Categories**: Cat-theories represent "big-step" operational semantics:*
  (* Often this means a rewrite to a *normal form*. We use the term more generally.)
  — a composite of morphisms is a *big-step* rewrite.
- **Posets**: Pos-theories represent "full-step" operational semantics:
  — a hom-poset boolean is the *existence* of a big-step rewrite.
- **Sets**: Set-theories represent denotational semantics:
  — a hom-set element is 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 can use labelled graphs to incorporate important concepts such as bisimulation; see §8.

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 [20], 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:



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: Gph → Cat maps small-step operational semantics to big-step operational semantics.
- Change of base along FP: Cat → Pos maps big-step operational semantics to full-step operational semantics.
- Change of base along US: Pos → Set maps full-step operational semantics to denotational semantics.

In §7 we mention that models of all V-theories for all enriching V 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, illustrating the idea of studying formal 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" [35]. 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 Corporation. We also thank Richard Garner, Todd Trimble and others at the n-Category Café for classifying cartesian closed categories where every object is a finite coproduct of copies of the terminal object [2].

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:

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

Lawvere theories formalize this idea. For example, there is a Lawvere theory Th(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 [5, Chap. 4]; for a more computer-science-oriented approach see Crole [10, Chap. 3]. In §4 we see that this construction is actually given by the Cat-theory for categories with finite products.

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 [14]. In 1965, Linton [19] proved that Lawvere theories correspond to "finitary monads" on the category of sets. For every Lawvere theory T, there is an adjunction:

$$\mathsf{Set} \underset{U}{\overset{F}{\rightleftarrows}} \mathsf{Mod}(\mathsf{T}, \mathsf{Set}). \qquad (\perp)$$

The functor

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

sends each model μ to its underlying 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 $n \in \mathsf{N}$ to the representable functor $\mathsf{T}(\tau(n), -)\colon \mathsf{T} \to \mathsf{Set}$, and in general any set X to the colimit of all such representables as n ranges over the poset of finite subsets of X. In rough terms, $F(X)$ is the model of all n-ary operations from T on the set X.

If we momentarily abbreviate $\mathsf{Mod}(\mathsf{T}, \mathsf{Set})$ as Mod, we obtain an adjunction

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

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. This forms the set of all terms that can be constructed from applying the operations to the elements, subject to the equations of the theory. The monad constructed this way is always **finitary**: that is, it preserves filtered colimits [1], or its action on sets is determined by its action on finite sets.

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 k_T. To see what this is doing, note that:

$$\mathsf{Kl}(T)^{\mathrm{op}}(n, m) = \mathsf{Kl}(T)(m, n) = \mathsf{Set}(m, T(n))$$

where the latter is considered as m n-ary operations in the Lawvere theory k_T. When T is finitary, the monad arising from this Lawvere theory is naturally isomorphic to T itself.

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 $\mathsf{Mod}(\mathsf{T})$ 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]. For more details see [5, 18, 24].

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 [16] and replace sets with objects of a cartesian closed category V, called the "enriching" category or "base". A V-enriched category or **V-category** C is:

a collection of objects	$\mathrm{Ob}(\mathsf{C})$
a hom-object function	$\mathsf{C}(-,-) \colon \mathrm{Ob}(\mathsf{C}) \times \mathrm{Ob}(\mathsf{C}) \to \mathrm{Ob}(\mathsf{V})$
composition morphisms	$\circ_{a,b,c} \colon \mathsf{C}(b,c) \times \mathsf{C}(a,b) \to \mathsf{C}(a,c) \quad \forall a, b, c \in \mathrm{Ob}(\mathsf{C})$
identity-assigning morphisms	$i_a \colon 1_{\mathsf{V}} \to \mathsf{C}(a,a) \quad \forall a \in \mathrm{Ob}(\mathsf{C})$

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

a function	$F \colon \mathrm{Ob}(\mathsf{C}) \to \mathrm{Ob}(\mathsf{D})$
a collection of morphisms	$F_{ab} \colon \mathsf{C}(a,b) \to \mathsf{D}(F(a), F(b)) \quad \forall a, b \in \mathsf{C}$

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

$$\text{a family} \quad \alpha_a \colon 1_{\mathsf{V}} \to \mathsf{D}(F(a), G(a)) \quad \forall a \in \mathrm{Ob}(\mathsf{C})$$

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

We can construct new V-categories from old by taking products and opposites in an obvious way. There is also a V-category denoted $\underline{\mathsf{V}}$ with the same objects as V and with hom-objects given by the internal hom:

$$\underline{\mathsf{V}}(v, w) = w^v \quad \forall v, w \in \mathsf{V}.$$

The concepts of adjunction and monad generalize straightforwardly to V-categories, and when we speak of an adjunction or monad in the enriched context this generalization is what we mean [16]. For example, there is an adjunction

$$\underline{\mathsf{V}}(u \times v, w) \cong \underline{\mathsf{V}}(u, w^v)$$

called "currying"; its counit is evaluation.

We can generalize products and coproducts to the enriched context. Given a V-category C, a **V-product** of an n-tuple of objects $b_1, \ldots, b_n \in \mathrm{Ob}(C)$ is an object b equipped with V-natural isomorphism

$$\mathsf{C}(-, b) \cong \prod_{i=1}^{n} \mathsf{C}(-, b_i). \tag{2}$$

If such an object b exists, we denote it by $\prod_{i=1}^n b_i$. This makes sense even when $n = 0$: a 0-ary product in C is called a **V-terminal object** and denoted as 1_{C}.

Whenever V is cartesian closed, the finite products in V are also V-products in $\underline{\mathsf{V}}$; this mainly amounts to saying

$$(u \times v)^w \cong u^w \times v^w \quad \text{and} \quad 1_{\mathsf{V}}^w \cong 1_{\mathsf{V}}.$$

Conversely, any finite V-product in V is also a product in the usual sense. In a general V-category C, it makes no sense to say a V-product is a product in the usual sense. However, the V-natural isomorphism in Eq. (2) gives rise to a morphism

$$\pi_i \colon 1_{\mathsf{V}} \to \mathsf{C}(b, b_i)$$

for each i, defined as the composite

$$1_{\mathsf{V}} \xrightarrow{i_a} \mathsf{C}(b, b) \xrightarrow{\sim} \prod_{i=1}^{n} \mathsf{C}(b, b_i) \longrightarrow \mathsf{C}(b, b_i).$$

These morphisms π_i serve as substitutes for the projections from b to b_i. They are "elements" of the hom-objects $\mathsf{C}(b, b_i)$, where an **element** of $v \in \mathrm{Ob}(\mathsf{V})$ is a morphism from 1_{V} to v. Elements of hom-objects behave much like morphisms for C. For example, we can define a morphism

$$p_i \colon \mathsf{C}(a, b) \to \mathsf{C}(a, b_i),$$

which acts like composition with π_i as follows:

$$\mathsf{C}(a, b) \xrightarrow{\sim} 1_{\mathsf{V}} \times \mathsf{C}(a, b) \xrightarrow{\pi_i \times 1} \mathsf{C}(b, b_i) \times \mathsf{C}(a, b) \xrightarrow{\circ_{a,b,b_i}} \mathsf{C}(a, b_i).$$

This morphism is V-natural in a, and the isomorphism in Eq. (2) has components given by the morphisms p_i.

Similarly, the **V-coproduct** of $b_1, \ldots, b_n \in \mathrm{Ob}(C)$ is an object b equipped with a V-natural isomorphism

$$C(b, -) \cong \prod_{i=1}^{n} C(b_i, -).$$

If such an object exists, we denote it by $\sum_{i=1}^n b_i$. When $n = 0$ we call this a V**-initial object** 0_{C}. When V is cartesian closed, any finite coproduct that exists in V is also a V-coproduct in $\underline{\mathsf{V}}$; this mainly amounts to saying

$$u^{v+w} \cong u^v \times u^w \quad \text{and} \quad 0^w \cong 1_{\mathsf{V}}$$

whenever 0 is an initial object of V. Conversely, any finite V-coproduct that exists in V is also a coproduct in the usual sense.

We say that a V-functor $F\colon \mathsf{C} \to \mathsf{D}$ **preserves** V-products if for every $b = \prod_{i=1}^n b_i$ in C, the V-natural transformations

$$F(\pi_i \circ -)\colon \mathsf{D}(-, F(b)) \to \mathsf{D}(-, F(b_i))$$

are the components of a V-natural isomorphism

$$\mathsf{D}(-, F(b)) \cong \prod_{i=1}^n \mathsf{D}(-, F(b_i)),$$

and similarly for V-coproducts.

A bit more subtly, generalizing the product and internal hom of V, a V-category C can have "tensors" and "powers" (which are sometimes called "copowers" and "cotensors"). Given $a \in \mathrm{Ob}(\mathsf{C})$ and $v \in \mathrm{Ob}(\mathsf{V})$, we say an object $v \cdot a \in \mathrm{Ob}(\mathsf{C})$ is the **tensor** of a by v if it is equipped with isomorphisms

$$\mathsf{C}(v \cdot a, b) \cong \mathsf{C}(a, b)^v$$

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:

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

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

$$\mathsf{C}(a, b^v) \cong \mathsf{C}(a, b)^v$$

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

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

When all objects of C have both tensors and powers by some object $v \in \mathsf{V}$, we have isomorphisms

$$\mathsf{C}(v \cdot a, b) \cong \mathsf{C}(a, b)^v \cong \mathsf{C}(a, b^v)$$

that are V-natural in a and b. When V is cartesian closed and $\mathsf{C} = \underline{\mathsf{V}}$ this holds for all $v \in \mathsf{V}$, since we can take $v \cdot a$ to be the product in V and take b^v to be the exponential.

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.

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" [30]. The original definition is as follows: for a symmetric monoidal closed category $(\mathsf{V}, \otimes, 1)$, 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

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

that preserves these powers. A "model" of a V-theory is a V-functor $\mu \colon \mathsf{T} \to \mathsf{V}$ that preserves powers by finite objects of V. 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 the enriched setting.

However, this sort of V-enriched Lawvere theory has arities for every finite object of V. These *generalized arities* may be very powerful—rather than only inputting n-tuples of terms, we can input any finite object of terms. Despite its potential, this generalization remains largely unexploited in computer science. Power [17] introduced "enriched sketches" as a way of presenting enriched Lawvere theories, but to the authors' knowledge these are not yet widely understood or used. What does it mean for an operation to take in a finite graph of terms? How can we learn to use this generality? One clue that we note in Example 7 is that limits and colimits are operations whose arity is a "shape" rather than only a natural number. We hope that this idea is explored more widely, so that we can use more general arities in both mathematics and computer programming.

In this paper, however, we only consider *natural number* arities, while still retaining enrichment. To do this we use the work of Lucyshyn-Wright [20], who along with Power [27] has generalized Power's original ideas to allow a more flexible choice of arities. We also limit ourselves to the case where the tensor product of V is cartesian. This has a significant simplifying effect, yet it suffices for many cases of interest in computer science.

Thus, in all that follows, we let $(\mathsf{V}, \times, 1_{\mathsf{V}})$ be a cartesian closed category equipped with chosen finite coproducts of the terminal object 1_{V}, say

$$n_{\mathsf{V}} = \sum_{i \in n} 1_{\mathsf{V}}.$$

Define N_{V} to be the full subcategory of V containing just these objects n_{V}. There is also a V-category $\underline{\mathsf{N}}_{\mathsf{V}}$ whose objects are those of N_{V} and whose hom-objects are given as in V. We define the V-category of **arities** for V to be

$$\mathsf{A}_{\mathsf{V}} := \underline{\mathsf{N}}_{\mathsf{V}}^{\mathrm{op}}.$$

We shall soon see that A_{V} has finite V-products.

Definition 1. We define a **V-theory** (T, τ) to be a V-category T equipped with a V-functor

$$\tau \colon \mathsf{A}_{\mathsf{V}} \to \mathsf{T}$$

that is bijective on objects and preserves finite V-products.

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

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

that preserves finite V-products.

Just as all the objects of a Lawvere theory are finite products of a single object, we shall see that all the objects of T are finite V-products of the object

$$t = \tau(1_{\mathsf{V}}).$$

Definition 3. For every V-theory (T, τ) and every V-category C with finite V-products, we define $\mathsf{Mod}(\mathsf{T}, \mathsf{C})$, the **category of models** of (T, τ) in C, to be the category for which an object is a V-functor $\mu \colon \mathsf{T} \to \mathsf{C}$ that preserves finite V-products and a morphism is a V-natural transformation.

Definition 4. We define VLaw, the **category of V-theories**, to be the category for which an object is a V-theory and a morphism from (T, τ) to (T', τ') is a V-functor $f \colon \mathsf{T} \to \mathsf{T}'$ that preserves finite V-products and has $f\tau = \tau'$.

The basic monadicity results for Lawvere theories generalize to V-theories when V is complete and cocomplete, as in the main examples we consider: $\mathsf{V} = \mathsf{Gph}, \mathsf{Cat}, \mathsf{Pos}$, and Set. Under this extra assumption $\mathsf{Mod}(\mathsf{T}, \mathsf{C})$ naturally becomes a V-category $\underline{\mathsf{Mod}}(\mathsf{T}, \mathsf{C})$. Furthermore, there is a V-functor

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

sending any model $\mu \colon \mathsf{T} \to \mathsf{V}$ to its underlying object $\mu(t) \in \mathsf{V}$. Recall that monads and adjunctions make sense in VCat, just as they do in Cat. The V-functor U has a left adjoint

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

and $\underline{\mathsf{Mod}}(\mathsf{T}, \mathsf{V})$ is equivalent to the V-category of algebras of the resulting monad $T = UF$. More precisely:

Theorem 5. Suppose V is cartesian closed, complete and cocomplete, and has chosen finite coproducts of the terminal object. Let (T, τ) be a V-theory. Then there is a monadic adjunction

$$\underline{\mathsf{V}} \underset{U}{\overset{F}{\rightleftarrows}} \underline{\mathsf{Mod}}(\mathsf{T}, \mathsf{V}). \qquad (\perp)$$

Proof. This follows from Lucyshyn-Wright's general theory [20], so our task is to explain how. He allows V to be a symmetric monoidal category, and uses a more general concept of algebraic theory with a system of arities given by any fully faithful symmetric monoidal V-functor $j \colon \mathsf{J} \to \underline{\mathsf{V}}$. For us $\mathsf{J} = \underline{\mathsf{N}}_{\mathsf{V}}$ and $j \colon \underline{\mathsf{N}}_{\mathsf{V}} \to \underline{\mathsf{V}}$ is the obvious inclusion; this is his Example 3.7.

Lucyshyn-Wright defines a **J-theory** to be a V-functor $\tau \colon \mathsf{J}^{\mathrm{op}} \to \mathsf{T}$ that is the identity on objects and preserves powers by objects in J (or more precisely, their images under j). For us $\mathsf{J}^{\mathrm{op}} = \mathsf{A}_{\mathsf{V}}$. We are only demanding that $\tau \colon \mathsf{A}_{\mathsf{V}} \to \mathsf{T}$ be bijective on objects, but we can make it the identity on objects simply by renaming the objects of T. So, to apply his theory, we need to show that a V-functor $\tau \colon \mathsf{A}_{\mathsf{V}} \to \mathsf{T}$ preserves powers by objects in N_{V} if and only if it preserves finite V-products. This is Lemma 12 below.

He defines a model (or "algebra") of a J-theory to be a V-functor $\tau \colon \mathsf{T} \to \underline{\mathsf{V}}$ that preserves powers by objects in J. He defines a morphism of models to be a V-natural transformation between such V-functors. So, to apply his theory, we also need to show that when $\mathsf{J} = \underline{\mathsf{N}}_{\mathsf{V}}$, a V-functor $\mu \colon \mathsf{T} \to \underline{\mathsf{V}}$ preserves powers by objects of J if and only if it preserves finite V-products. This is Lemma 13 below.

A technical concept fundamental to Lucyshyn-Wright's theory is that of an **eleutheric** system of arities $j \colon \mathsf{J} \to \underline{\mathsf{V}}$. This is one where the left Kan extension of any V-functor $T \colon \mathsf{J} \to \underline{\mathsf{V}}$ along j exists

and is preserved by each V-functor $\underline{\mathsf{V}}(x,-)\colon \underline{\mathsf{V}} \to \underline{\mathsf{V}}$. In Example 7.5.5 he shows that $j\colon \underline{\mathsf{N}}_{\mathsf{V}} \to \underline{\mathsf{V}}$ is eleutheric when V is countably cocomplete. In Thm. 8.9 shows that when $j\colon \mathsf{J} \to \underline{\mathsf{V}}$ is eleutheric, and has equalizers, we may form the V-category $\underline{\mathsf{Mod}}(\mathsf{T},\mathsf{V})$, and that the forgetful V-functor

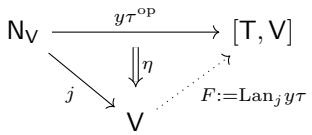
$$U\colon \underline{\mathsf{Mod}}(\mathsf{T},\mathsf{V}) \to \underline{\mathsf{V}}$$

is monadic. This is the result we need. So, our theorem actually holds whenever V is cartesian closed, with equalizers and countable colimits, and has chosen finite coproducts of the initial object. □

Before turning to examples, a word about Lucyshyn-Wright's construction of the left adjoint F and the monad T is in order. These rely on the "free model" on an object $n_{\mathsf{V}} \in \mathsf{V}$. This is the enriched generalization of the free model described in Section 2: it is the composite of $\tau^{\mathrm{op}}\colon \mathsf{A}_{\mathsf{V}}^{\mathrm{op}} \to \mathsf{T}^{\mathrm{op}}$ with the enriched Yoneda embedding $y\colon \mathsf{T}^{\mathrm{op}} \to [\mathsf{T},\mathsf{V}]$:

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

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 this "free model" functor $y\tau^{\mathrm{op}}$ to all of V is specified by a somewhat higher-powered generalization: it is the left Kan extension of $y\tau$ along j.



This is the universal "best solution" to the problem of making the triangle commute up to a V-natural transformation. That is, for any functor $G\colon \mathsf{V} \to [\mathsf{T},\mathsf{V}]$ and V-natural transformation $\theta\colon y\tau \Rightarrow Gj$, the latter factors uniquely through η. From the adjunction between V and the category of models $\mathsf{Mod}(\mathsf{T},\mathsf{V})$ we obtain a V-enriched monad

$$T = UF\colon \mathsf{V} \to \mathsf{V},$$

and this has a more concrete formula as an enriched coend:

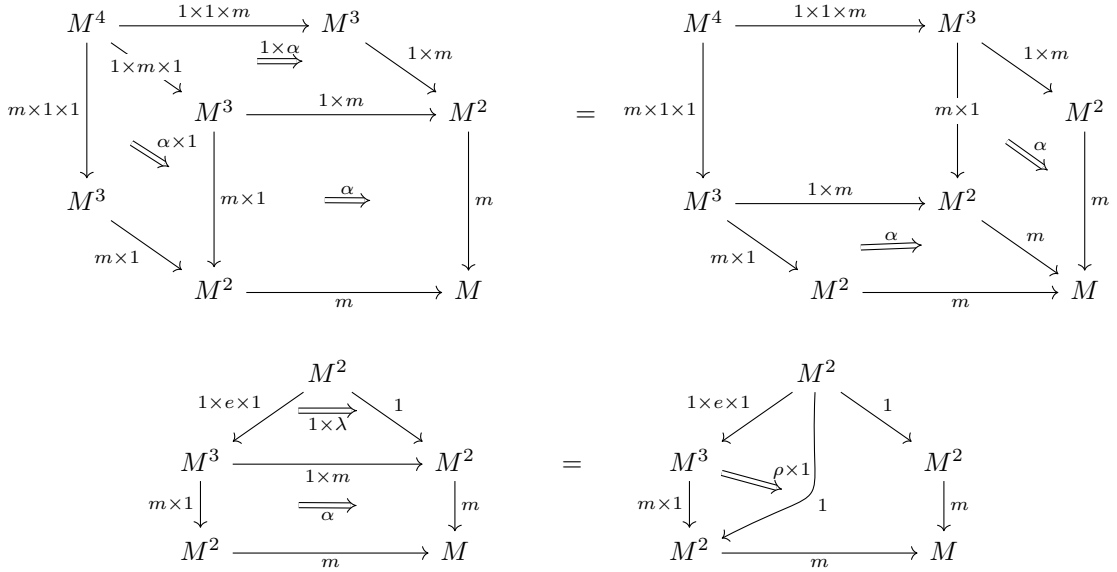
$$T(V) = \int^{n_{\mathsf{V}} \in \mathsf{N}_{\mathsf{V}}} V^{n_{\mathsf{V}}} \times \mathsf{T}(t^{n_{\mathsf{V}}}, t).$$

We next give two examples of a rather abstract nature, where we show how Cat-enriched Lawvere theories can describe categories with extra structure. In Section 8 we study examples more directly connected to operational semantics.

Example 6. When $\mathsf{V} = \mathsf{Cat}$, a V-category is a 2-category, so a V-theory deserves to be called a **2-theory**. For example, let $\mathsf{T} = \mathsf{Th}(\mathrm{PsMon})$ be the 2-theory of pseudomonoids [11]. A pseudomonoid is a weakened version of a monoid: rather than associativity and unitality *equations*, it has 2-isomorphisms called the associator and unitors, which we can treat as *rewrite rules*. To equate various possible rewrite sequences, these 2-isomorphisms must obey equations called "coherence laws". Here is a presentation of the 2-theory for pseudomonoids:

$\mathsf{Th}(\mathrm{PsMon})$

sort	M	pseudomonoid
operations	$m\colon M^2 \to M$	multiplication
	$e\colon 1 \to M$	identity
rewrites	$\alpha\colon m \circ (m \times \mathrm{id}_M) \xRightarrow{\sim} m \circ (\mathrm{id}_M \times m)$	associator
	$\lambda\colon m \circ (e \times \mathrm{id}_M) \xRightarrow{\sim} \mathrm{id}_M$	left unitor
	$\rho\colon m \circ (\mathrm{id}_M \times e) \xRightarrow{\sim} \mathrm{id}_M$	right unitor
equations		



We write the equations as commutative diagrams merely for convenience; they could also be written as equations in a more traditional style. The top diagram expresses the pentagon identity for the associator, while the bottom one expresses the usual coherence law involving the left and right unitors.

Models of $\mathsf{T} = \mathsf{Th}(\mathrm{PsMon})$ in Cat are monoidal categories: let us explore this example in more detail. A model of T is a finite-product-preserving 2-functor $\mu\colon \mathsf{T} \to \mathsf{Cat}$, which sends

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

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 associative and unital up to natural isomorphism; so these models are precisely monoidal categories.

Given two models $\mu, \nu \colon \mathsf{T} \to \mathsf{Cat}$, a morphism of models is a 2-natural transformation $\varphi \colon \mu \Rightarrow \nu$; this amounts to a strict monoidal functor $\varphi \colon (\mathsf{C}, \otimes_C, I_C) \to (\mathsf{D}, \otimes_D, I_D)$. The strictness arises because morphisms between models are 2-natural transformations rather than pseudonatural transformations. There is a substantial amount of theory on pseudomonads and pseudoalgebras [8, 12], but to the authors' knowledge the theory-monad correspondence has not yet been extended to weak enrichment.

Finally, because Cat is complete and cocomplete, the category of models $\mathsf{Mod}(\mathsf{T}, \mathsf{Cat})$ can be promoted to a 2-category $\underline{\mathsf{Mod}}(\mathsf{T}, \mathsf{Cat})$. This is the 2-category of monoidal categories, strict monoidal functors, and monoidal natural transformations.

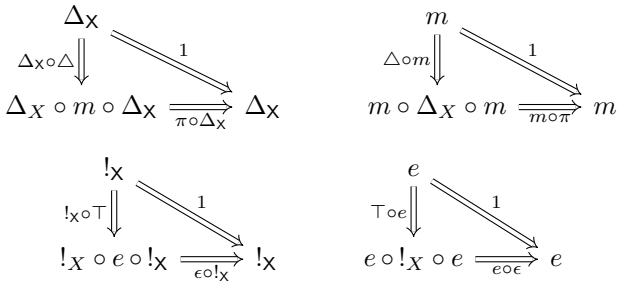
We can accomplish the same thing on the monad side: a Cat-enriched monad is called a **2-monad**, and T gives rise to the "free monoidal category" 2-monad T on Cat [8]. To apply this 2-monad to $C \in \mathsf{Cat}$ we first form the free model on C by taking a left Kan extension as above, and then evaluate this model at the generating object. 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, $T(\mathsf{C})$ is the monoidal category consisting of all finite tensor products of objects of C and all morphisms built from those of C by composition and tensoring together with associators and unitors obeying the necessary coherence laws. Morphisms of these algebras are strict monoidal functors, while 2-morphisms are natural transformation. We thus have a 2-equivalence between $\underline{\mathsf{Mod}}(\mathsf{T}, \mathsf{Cat})$ and the 2-category of algebras of T.

In this way, 2-theories generalize equipping *set*-like objects with operations obeying equations to equipping *category*-like objects with operations obeying equations up to transformations that obey equations of their own. In particular, this gives us a way to present graphical calculi such as string diagrams – the language of monoidal categories.

Example 7. Enrichment generalizes operations in more ways than by weakening equations to coherent isomorphisms. We can also use 2-theories to describe other structures that make sense inside 2-categories, such as adjunctions.

For example, we may define a cartesian category X to be one equipped with right adjoints to the diagonal $\Delta_{\mathsf{X}} \colon \mathsf{X} \to \mathsf{X} \times \mathsf{X}$ and the unique functor $!_{\mathsf{X}} \colon \mathsf{X} \to 1_{\mathsf{Cat}}$. These right adjoints are a functor $m \colon \mathsf{X}^2 \to \mathsf{X}$ describing binary products in X and a functor $e \colon 1 \to \mathsf{X}$ picking out the terminal object in X. We can capture the fact that they are right adjoints by providing them with units and counits and imposing the triangle equations. There is thus a 2-theory $\mathsf{Th}(\mathsf{Cart})$ whose models in Cat are categories with chosen finite products. More generally a model of this 2-theory in any 2-category C with finite products is called a **cartesian object** in C.

		$\mathsf{Th}(\mathsf{Cart})$
type	X	cartesian object
operations	$m \colon \mathsf{X}^2 \to \mathsf{X}$	product
	$e \colon 1 \to \mathsf{X}$	terminal element
rewrites	$\triangle \colon \mathrm{id}_{\mathsf{X}} \Longrightarrow m \circ \Delta_{\mathsf{X}}$	unit of adjunction between m and Δ_{X}
	$\pi \colon \Delta_{\mathsf{X}} \circ m \Longrightarrow \mathrm{id}_{\mathsf{X}^2}$	counit of adjunction between m and Δ_{X}
	$\top \colon \mathrm{id}_{\mathsf{X}} \Longrightarrow e \circ !_{\mathsf{X}}$	unit of adjunction between e and $!_{\mathsf{X}}$
	$\epsilon \colon !_{\mathsf{X}} \circ e \Longrightarrow \mathrm{id}_1$	counit of adjunction between e and $!_{\mathsf{X}}$
equations		



Again we write the equations as commutative diagrams, but this time commutative triangles of 2-morphisms in $\mathsf{Th}(\mathsf{Cart})$. These are the triangle equations that force m to be the right adjoint of Δ_{X} and e to be the right adjoint of $!_{\mathsf{X}}$. A model of $\mathsf{Th}(\mathsf{Cart})$ is a category with chosen binary products and a chosen terminal object; morphisms in $\mathsf{Mod}(\mathsf{Th}(\mathsf{Cart}), \mathsf{Cat})$ are functors that strictly preserve this extra structure.

The subtle interplay between the cartesian structure of $\mathsf{Th}(\mathsf{Cart})$ and the cartesian structure of the object $\mathsf{X} \in \mathsf{Th}(\mathsf{Cart})$ is an example of the "microcosm principle": objects with a given structure are most generally defined in a context that has the same sort of structure. As seen in the previous example, we can also define pseudomonoids in any 2-category with finite products, but this is excess to requirements: one can in fact define them more generally in any monoidal 2-category [11].

In fact, if we let arities be finite categories, we would have Cat-theories of categories with finite limits and colimits. However, for the purposes of this paper we are using only natural number arities. This suffices for constructing $\mathsf{Th}(\mathsf{Cart})$ and also $\mathsf{Th}(\mathsf{CoCart})$, the theory of categories with chosen binary coproducts and a chosen initial object. Various other kinds of categories—distributive categories, rig categories, etc.—can also be expressed using Cat-theories with natural number arities. This gives a systematic formalization of these categories, internalizes them to new contexts, and allows for the generation of 2-monads that describe them.

5. Natural Number Arities

In this section we prove the lemmas required for Theorem 5 and our study of base change in Section 6. Throughout this section V is cartesian closed with chose n-fold coproducts n_{V} of its terminal object.

We begin with a study of N_{V}, the full subcategory of V on the objects n_{V}. First we must resolve a potential ambiguity. On the one hand, for any object b of V we can form the exponential $b^{n_{\mathsf{V}}}$. On the other hand, we can take the product of n copies of b, which we call b^n. Luckily these are the same, or at least naturally isomorphic:

Lemma 8. The functors $(-)^{n_{\mathsf{V}}} \colon \mathsf{V} \to \mathsf{V}$ and $(-)^n \colon \mathsf{V} \to \mathsf{V}$ are naturally isomorphic.

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

$$\begin{aligned}
\mathsf{V}(a, b^{n_{\mathsf{V}}}) &\cong \mathsf{V}(a \times n_{\mathsf{V}}, b) && \text{hom-tensor adjunction} \\
&= \mathsf{V}(a \times (n \cdot 1_{\mathsf{V}}), b) && \text{definition of } n_{\mathsf{V}} \\
&\cong \mathsf{V}(n \cdot (a \times 1_{\mathsf{V}}), b) && \text{products distribute over coproducts} \\
&\cong \mathsf{V}(n \cdot a, b) && \text{unitality} \\
&\cong \mathsf{V}(a, b)^n && \text{definition of coproduct} \\
&\cong \mathsf{V}(a, b^n) && \text{definition of product.}
\end{aligned}$$

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

We can now understand coproducts, products and exponentials in N_V:

Lemma 9. If V is any cartesian closed category with chosen coproducts of the initial object then N_V is cartesian closed, with finite coproducts. The unique initial object of N_V is 0_V. The binary coproducts in N_V are unique, given by

$$m_\mathsf{V} + n_\mathsf{V} = (m+n)_\mathsf{V}.$$

The unique terminal object of N_V is 1_V, and the binary products are unique, given by

$$m_\mathsf{V} \times n_\mathsf{V} = (mn)_\mathsf{V}.$$

Exponentials in N_V are also unique, given by

$$m_\mathsf{V}{}^{n_\mathsf{V}} = (m^n)_\mathsf{V}.$$

Proof. In V we know that 0_V is an initial object and 1_V is a terminal object, by definition. Since the subcategory N_V is skeletal 0_V is the unique initial object and 1_V is the unique terminal object in N_V. Similarly, in V we have defined $(m+n)_\mathsf{V}$ to be a coproduct of m_V and n_V, so in N_V it is the unique such, and we can unambiguously write

$$m_\mathsf{V} + n_\mathsf{V} = (m+n)_\mathsf{V}.$$

Products distribute over coproducts in any cartesian closed category, so in V we have

$$m_\mathsf{V} \times n_\mathsf{V} \cong (1_\mathsf{V} + \cdots + 1_\mathsf{V}) \times (1_\mathsf{V} + \cdots + 1_\mathsf{V}) \cong (mn)_\mathsf{V}$$

where in the second step we use the distributive law twice. It follows that N_V has finite products, and since this subcategory is skeletal they are unique, given by

$$m_\mathsf{V} \times n_\mathsf{V} = (mn)_\mathsf{V}.$$

Finally, by Lemma 8 we have

$$m_\mathsf{V}{}^{n_\mathsf{V}} \cong m_\mathsf{V}^n \cong \prod_{i=1}^{n} m_\mathsf{V} \cong (m^n)_\mathsf{V}.$$

It follows that N_V has exponentials, and since this subcategory is skeletal they are unique, given by

$$m_\mathsf{V}^{n_\mathsf{V}} = (m^n)_\mathsf{V}. \qquad \square$$

We warn the reader that $\hom(m_\mathsf{V}, n_\mathsf{V})$ may not have n^m elements. It does in $\mathsf{Gph}, \mathsf{Cat}, \mathsf{Pos}$ and of course Set, but not in $\mathsf{V} = \mathsf{Set}^k$, where $|\hom(m_\mathsf{V}, n_\mathsf{V})| = n^{km}$. In fact, whenever N_V has finite hom-sets it is equivalent to FinSet^k for some k. The reason is that 2_V is an internal Boolean algebra in V, so its set of elements $\hom(1_\mathsf{V}, 2_\mathsf{V})$ must be some Boolean algebra B in Set. A further argument due to Garner and Trimble shows that N_V is completely characterized, up to equivalence, by this Boolean algebra, and any Boolean algebra can occur [2]. If this Boolean algebra is finite it must be isomorphic to $\{0,1\}^k$ for some $k \geq 0$. In this case, N_V is equivalent to FinSet^k.

Now suppose C is a V-category. The question arises whether the power of an object $c \in \mathsf{C}$ by n_V must also be the V-product of n copies of c. The answer is yes:

Lemma 10. Let C be a V-category and $c \in \mathrm{Ob}(\mathsf{C})$. Then the power c^{n_V} exists if and only if the n-fold V-product c^n exists, in which case they are isomorphic.

Proof. If the power $c^{n_{\mathsf{V}}}$ exists, we have

$$\begin{array}{rcll} \mathsf{C}(a, c^{n_{\mathsf{V}}}) & \cong & \mathsf{C}(a,c)^{n_{\mathsf{V}}} & \text{definition of powers} \\ & \cong & \mathsf{C}(a,c)^n & \text{Lemma 8} \end{array}$$

Each of these isomorphisms is V-natural in a, so $c^{n_{\mathsf{V}}}$ is the n-fold product of copies of c. Conversely, if the n-fold product c^n exists, we have

$$\begin{array}{rcll} \mathsf{C}(a, c^n) & \cong & \mathsf{C}(a,c)^n & \text{definition of } \mathsf{V}\text{-products} \\ & \cong & \mathsf{C}(a,c)^{n_{\mathsf{V}}} & \text{Lemma 8} \end{array}$$

and each of these isomorphisms is V-natural in a, so c^n is the power of c by n_{V}. □

To be precise, we need not only to define preservation of V-products but that of V-powers, as that is in the actual definition of V-theories and their models. We then show that they are equivalent.

Let C be a V-category which is powered over V. Similarly to V-products, the ******

Lemma 11. Suppose C is a V-category such that every object is the n-fold V-product c^n of some object c. Then a V-functor $F\colon \mathsf{C} \to \mathsf{D}$ preserves finite V-products if and only if it preserves N_{V}-powers.

Proof. Suppose F preserves finite V-products. Then for any object $b \in \mathrm{Ob}(\mathsf{C})$ there are isomorphisms

$$\begin{array}{rcll} F(b^{n_{\mathsf{V}}}) & \cong & F(b^n) & \text{Lemma 10} \\ & \cong & F(b)^n & \text{preservation of } \mathsf{V}\text{-products} \\ & \cong & F(b)^{n_{\mathsf{V}}} & \text{Lemma 10} \end{array}$$

so F preserves N_{V}-powers. Conversely suppose F preserves N_{V}-powers. To show that F preserves finite V-products it suffices to show it preserves the terminal object and binary products. For the former note that

$$\begin{array}{rcll} F(1_{\mathsf{C}}) & \cong & F(c^0) & \mathsf{V}\text{-terminal object is a 0-ary product} \\ & \cong & F(c^{0_{\mathsf{V}}}) & \text{Lemma 10} \\ & \cong & F(a)^{0_{\mathsf{V}}} & \text{preservation of } \mathsf{N}_{\mathsf{V}}\text{-powers} \\ & \cong & F(a)^0 & \text{Lemma 10} \\ & \cong & 1_{\mathsf{D}} & \mathsf{V}\text{-terminal object is a 0-ary product.} \end{array}$$

For the latter note that

$$\begin{array}{rcll} F(c^m \times c^n) & \cong & F(c^{m+n}) & \mathsf{V}\text{-products in } \mathsf{C} \\ & \cong & F(c^{(m+n)_{\mathsf{V}}}) & \text{Lemma 10} \\ & \cong & F(c)^{(m+n)_{\mathsf{V}}} & \text{preservation of } \mathsf{N}_{\mathsf{V}}\text{-powers} \\ & \cong & F(c)^{m+n} & \text{Lemma 10} \\ & \cong & F(c)^m \times F(c)^n & \text{definition of } m+n\text{-fold } \mathsf{V}\text{-product.} \end{array}$$

□

Lemma 12. Let V be cartesian closed with chosen finite coproducts of the terminal object and let T be a V-category. These conditions for a V-functor $\tau\colon \mathsf{A}_{\mathsf{V}} \to \mathsf{T}$ are equivalent:

(1) (T, τ) is a V-theory,
(2) τ preserves finite V-products,
(3) τ preserves powers by objects of N_{V}.

Proof. Conditions 1 and 2 are equivalent by definition. Since $\mathsf{A}_\mathsf{V} = \underline{\mathsf{N}}_\mathsf{V}^{\mathrm{op}}$, finite V-products in A_V are the same as finite V-coproducts in $\underline{\mathsf{N}}_\mathsf{V}$, which are the same as finite coproducts in N_V. Since every object in $\underline{\mathsf{N}}_\mathsf{V}$ is a finite coproduct of copies of 1_V, Lemma 11 implies that conditions 2 and 3 are equivalent. □

Lemma 13. Given a V-theory (T, τ) and a V-functor $\mu\colon \mathsf{T} \to \mathsf{C}$, the following conditions are equivalent:

- μ is a model of (T, τ),
- μ preserves finite V-products,
- μ preserves powers by objects of N_V.

Proof. Conditions 1 and 2 are equivalent by definition. Since τ is bijective on objects and preserves V-products each object of T is of the form t^n where $t = \tau(1_\mathsf{V})$. Thus, Lemma 11 implies that conditions 2 and 3 are equivalent. □

6. Change of Base

We now have the tools to formulate the main idea: a choice of enrichment for Lawvere theories corresponds to a choice 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.

Suppose that V and W are enriching categories of the sort we are considering: cartesian closed categories equipped with chosen finite coproducts of the terminal object. Suppose $F\colon \mathsf{V} \to W$ preserves finite products. This induces a **change of base** functor $F_*\colon \mathsf{VCat} \to \mathsf{WCat}$ [9] which takes any V-category C and produces a W-category $F_*(\mathsf{C})$ with the same objects but with

$$F_*(\mathsf{C})(a, b) := F(\mathsf{C}(a, b))$$

for all objects a, b. Composition in $F_*(\mathsf{C})$ is defined by

$$F(\mathsf{C}(b, c)) \times F(\mathsf{C}(a, b)) \xrightarrow{\sim} F(\mathsf{C}(b, c) \times \mathsf{C}(a, b)) \xrightarrow{F(\circ_{a,b,c})} F(C(a, b)).$$

The identity-assigning morphisms are given by

$$1 \xrightarrow{\sim} F(1) \xrightarrow{F(i_a)} F(\mathsf{C}(a, b)).$$

Moreover, if $f\colon \mathsf{C} \to \mathsf{D} \in \mathsf{VCat}$ is a V-functor, there is a W-functor $F_*(f)\colon F_*(C) \to F_*(D)$ that on objects equals f and on hom-objects equals $F(f)$. If $\alpha\colon f \Rightarrow g$ is a V-natural transformation and $c \in \mathrm{Ob}(\mathsf{C})$, then we define $F_*(\alpha)_c$ to be the composite

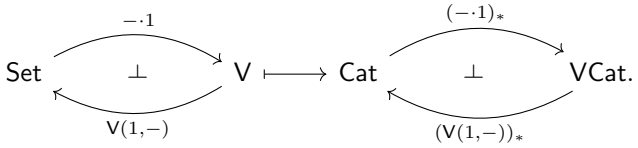
$$1 \xrightarrow{\sim} F(1) \xrightarrow{F(i_a)} F(\mathsf{C}(a, b)).$$

Thus, change of base actually gives a 2-functor from the 2-category of V-categories, V-functors and V-natural transformations to the corresponding 2-category for W.

In fact, the change of base operation gives a 2-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, if V has not just finite coproducts of the terminal object, but all coproducts of this object, there is a map of adjunctions



Each set X is mapped to the X-indexed coproduct of the terminal object in V 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 $\mathsf{C} \mapsto \underline{\mathsf{C}}$. 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 $\underline{\mathsf{N}}_\mathsf{V}$.

We now study how change of base affects theories and their models. We start by asking when a functor $F\colon \mathsf{V} \to \mathsf{W}$ induces a change of base $F_*\colon \mathsf{VCat} \to \mathsf{WCat}$ that "preserves enriched theories". That is, given a V-theory

$$\tau\colon \mathsf{A}_\mathsf{V} \to \mathsf{T}$$

we want to determine conditions for the base-changed functor

$$F_*(\tau)\colon F_*(\mathsf{A}_\mathsf{V}) \to F_*(\mathsf{T})$$

to induce a W-theory in a canonical way. Recall that we require V and W to be cartesian closed, equipped with chosen finite coproducts of their terminal objects. We thus expect the following conditions to be sufficient: F should be cartesian, and it should preserve the chosen finite coproducts of the terminal object:

$$F(n_\mathsf{V}) = n_\mathsf{W}$$

for all n.

Given these conditions there is a W-functor, in fact an isomorphism

$$\tilde{F}\colon \mathsf{A}_\mathsf{W} \to F_*(\mathsf{A}_\mathsf{V}).$$

On objects this maps n_W to n_V, and on hom-objects it is simply the identity from

$$\mathsf{A}_\mathsf{W}(m_\mathsf{W}, n_\mathsf{W}) = n_\mathsf{W}^{m_\mathsf{W}} = (n^m)_\mathsf{W}$$

to

$$F(\mathsf{A}_\mathsf{V}(m_\mathsf{V}, n_\mathsf{V})) = F(n_\mathsf{V}^{m_\mathsf{V}}) = F((n^m)_\mathsf{V}) = (n^m)_\mathsf{W}$$

where we use Lemma 9 in these computations.

Using this we obtain a composite W-functor

$$\mathsf{A}_\mathsf{W} \xrightarrow{\tilde{F}} F_*(\mathsf{A}_\mathsf{V}) \xrightarrow{F_*(\tau_\mathsf{V})} F_*(\mathsf{T}).$$

This is a bijection on objects and preserves finite V-products because each of the factors has these properties. It is thus a W-theory.

Theorem 14. Let V, W be cartesian closed categories with chosen finite coproducts of their terminal objects, and let $F\colon \mathsf{V} \to \mathsf{W}$ be a cartesian functor that preserves these chosen coproducts. Then F **preserves enriched theories**: that is, for every V-theory $\tau_{\mathsf{V}}\colon \mathsf{A}_{\mathsf{V}} \to \mathsf{T}$, the W-functor

$$\tau_{\mathsf{W}} := F_*(\tau_{\mathsf{V}}) \circ \tilde{F}\colon \mathsf{A}_{\mathsf{W}} \to F_*(\mathsf{T})$$

is a W-theory. Moreover, F preserves models: for every model $\mu\colon \mathsf{T} \to \mathsf{C}$ of $(\mathsf{T}, \tau_{\mathsf{V}})$, the W-functor $F_*(\mu)\colon F_*(\mathsf{T}) \to F_*(\mathsf{C})$ is a model of $(F_*(\mathsf{T}), \tau_{\mathsf{W}})$.

Proof. To show that $F_*(\mu)$ is a model, we need to show it preserves finite V-products. We do this separately for the V-terminal object and binary V-products. For the former, we need to show $F_*(\mu)1_{F*(\mathsf{T})}$ is W-terminal in $F_*(\mathsf{C})$, i.e. $F_*(\mathsf{C})(c, F_*(\mu)1_{F*(\mathsf{T})})$ is terminal in W for all objects c of C. But ... FINISH THIS!!!

□

Hence, any cartesian functor that preserves chosen finite coproducts of the terminal object gives 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 $(\mathsf{N}_{\mathsf{V}})^n$, the category of n-fold products of coproducts of 1_{V}, is also an eleutheric system of arities. In Section 8.4, we give an example of this 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. A morphism of V-theories $f\colon \mathsf{T} \to \mathsf{T}'$ 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, a V-product-preserving V-functor $g\colon \mathsf{C} \to \mathsf{C}'$ induces a "change-of-context" 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 $p_F\colon \int F \to \mathsf{D}$ given as follows:

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

For more details see [9, 15]. We noted in Section that VLaw and $\mathsf{Mod}(\mathsf{T}, \mathsf{C})$ are V-categories when V is complete and cocomplete: this and further conditions imply we can use the *enriched* Grothendieck construction [6]; 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 $\mathsf{N_V}$-powers and $\mathsf{N_V}$-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-context above.

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

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

consists of:

- a V-functor $f \colon \mathsf{T} \to \mathsf{T}'$ which preserves $\mathsf{N_V}$-powers,
- a V-functor $g \colon \mathsf{C} \to \mathsf{C}'$ which preserves $\mathsf{N_V}$-powers, and
- a V-natural transformation $\alpha \colon \mathsf{VMod}(f, g)(\mu) \to \mu'$.

Lemma 15. 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 16. There is a category $\mathrm{Thy} := \int \mathrm{thy}$ in which a morphism

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

is 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 lot of conceptual interaction. 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:

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

****

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 only 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-context.

8. Applications

In computer science literature, enriched algebraic theories have primarily been studied in the context of "computational effects" [29]. Mike Stay and Greg Meredith propose that enriched Lawvere theories can be utilized for the design of programming languages [36]. The primary difference between programming languages and enriched theories, however, is that the latter do not have a notion of *variable binding* – there has certainly been progress on this idea [13], but it leads us beyond our present focus. There are other approaches which instead use an enriched theory as a "compiler", which can translate a higher language with binding to one without – 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 [3].

Terms are constructed recursively by *variables*, *application*, and *abstraction*, and the basic rewrite is *beta reduction*, which substitutes the applied term for the bound variable:

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

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

Despite the apparent simplicity, there are complications regarding substitution. 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** [31], 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 a technique for programming languages to minimize the subtleties of variables. A great introduction into the strange world of combinators is given by Smullyan [34].

The insight of Stay and Meredith [35] is that even though Lawvere theories have no variables, through abstraction elimination a programming language can be made into an algebraic object. When representing a computational calculus as a 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)

type	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. (Here l, r denote the unitors and τ the symmetry of the product.) They are simply edges with vertices:

$$\begin{array}{rc} (((S -) =) \equiv)\colon & t^3 \xrightarrow{l^{-1}\times t^3} 1 \times t^3 \xrightarrow{S\times t^3} t^4 \xrightarrow{(- -)\times t^2} t^3 \xrightarrow{(- -)\times t} t^2 \xrightarrow{(- -)} t \\ \sigma\Downarrow & \| \qquad\qquad\qquad\qquad \Downarrow \qquad\qquad\qquad\qquad \| \\ ((- \equiv) (= \equiv))\colon & t^3 \xrightarrow{t^2\times\Delta} t^4 \xrightarrow{t\times\tau\times t} t^4 \xrightarrow{(- -)\times(- -)} t^2 \xrightarrow{(- -)} t \end{array}$$

$$\begin{array}{rc} ((K -) =)\colon & t^2 \xrightarrow{l^{-1}\times t^2} 1 \times t^2 \xrightarrow{K\times t^2} t^3 \xrightarrow{(- -)\times t} t^2 \xrightarrow{(- -)} t \\ \kappa\Downarrow & \| \qquad\qquad\qquad \Downarrow \qquad\qquad\qquad \| \\ -\colon & t^2 \xrightarrow{t\times !} t \times 1 \xrightarrow{r} t \end{array}$$

$$\begin{array}{rc} (I -)\colon & t \xrightarrow{l^{-1}} 1 \times t \xrightarrow{I\times t} t^2 \xrightarrow{(- -)} t \\ \iota\Downarrow & \| \qquad\qquad \Downarrow \qquad\qquad \| \\ -\colon & t \xrightarrow{\qquad\qquad t \qquad\qquad} t \end{array}$$

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

$$\begin{array}{rc} ((KS)I)\colon & 1 \xrightarrow{S\times I} t^2 \xrightarrow{((K -) =)} t \\ \kappa\circ(S\times I)\Downarrow & \| \qquad\qquad \Downarrow \qquad\qquad \| \\ S\colon & 1 \xrightarrow{S\times I} t^2 \xrightarrow{\quad - \quad} t \end{array}$$

(Morphisms $1 \to t$ are the "closed" terms, meaning they have no holes in which to substitute terms; in general morphisms $t^n \to t$ are terms with n holes, or n-ary operations, and the same reasoning applies.)

A model of this theory is a power-preserving Gph-functor $\mu\colon \mathsf{Th}(\mathsf{SKI}) \to \mathsf{Gph}$. 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):

$$((((S\ (K\ (I\ I)))\ S)\dots).$$

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,

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

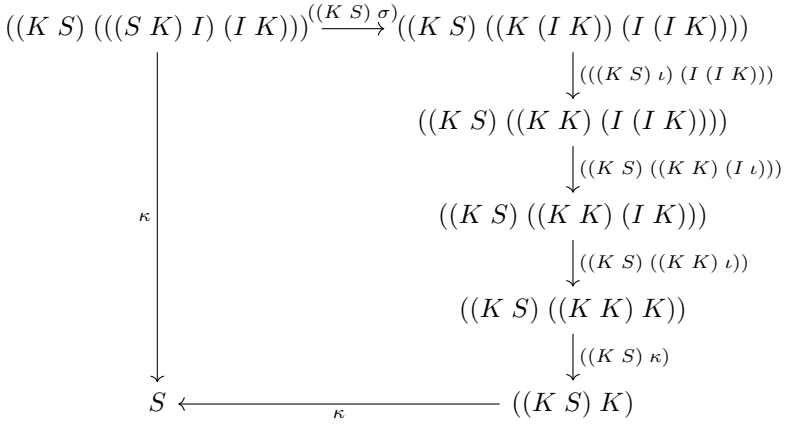
maps the "syntactic" graph of all closed terms and rewrites coherently into the "semantic" graph, meaning a rewrite in the theory $a \Rightarrow b$ is sent to a rewrite in the model $\mu(a) \Rightarrow \mu(b)$.

These rewrites in the image of μ are *graph transformations*, which are just like natural transformations of functors, without the commuting diagram: given two graph homomorphisms $f, g \colon G \to H$, a graph transformation $\alpha \colon f \Rightarrow g$ is a function $G_0 \to H_1$ which sends a vertex $v \in G$ to an edge $\alpha(v)$ with source $f(v)$ and target $g(v)$.

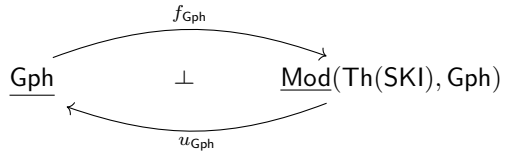
This is how μ realizes $\mathsf{Th}(\mathsf{SKI})$ as a graph of terms and rewrites: in the same way that a natural transformation of two constant functors $a \Rightarrow b \colon 1 \to \mathsf{C}$ is a morphism $a(1) \to b(1)$ in C, a rewrite of closed terms $a \Rightarrow b \colon 1 \to \mu(t)$ corresponds 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:



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 $\mathsf{Th}(\mathsf{SKI})$, 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 Thm. 5:



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

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

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:

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

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** operational semantics. The "free category" functor $\mathrm{FC}\colon \mathsf{Gph} \to \mathsf{Cat}$ 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

$$\begin{array}{rl} \text{objects} & \text{vertices of } G \\ \text{morphisms} & (v_1, e_1, v_2, e_2, ..., v_n) \ : \quad \forall i < n \ \ s(e_i) = v_i \ , \ t(e_i) = v_{i+1} \\ \text{composition} & (v_1, e_1, v_2, e_2, ..., v_n) \circ (v'_1, e'_1, v'_2, e'_2, ..., v'_n) = (v_1, e_1, ..., v_n = v'_1, e'_1, ..., v'_n) \end{array}$$

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

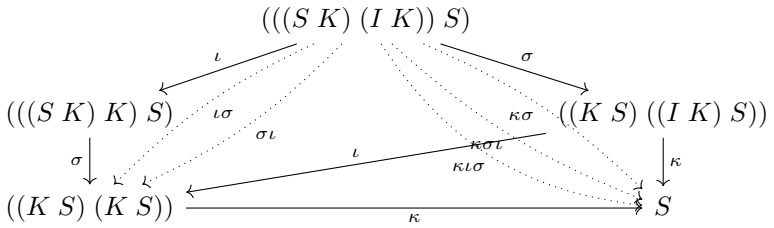
$$\mu^{\mathsf{Cat}}_{SKI} := \mathrm{FC}_*(\mu^{\mathsf{Gph}}_{SKI})$$

is the category which represents the big-step operational semantics of the SKI-calculus. To correpond with the conventional meaning of big-step semantics, as noted in §1, we can quotient by Curry's equations to identify SKI-terms which are extensionally equivalent [4].

The same reasoning applies to the "free poset" functor $\mathrm{FP}\colon \mathsf{Cat} \to \mathsf{Pos}$; 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 truth values, simply asserting the existence of a rewrite sequence between terms, without distinguishing between different paths. Starting from a free category, this is simply adding the property that all 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. We take the right adjoint $\mathrm{US}\colon \mathsf{Pos} \to \mathsf{Set}$ to the functor $\mathrm{FP}\colon \mathsf{Set} \to \mathsf{Pos}$ sending any set to the discrete poset on that set. The functor US collapses every connected component of the poset to a point. This extracts the denotational semantics of the language, by identifying all terms related by rewrites. If the rewrites are terminating and confluent, we can choose a representative term for each equivalence class: the unique term that admits no nontrivial rewrites.

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



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, US_* 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.

8.3. **Change-of-theory.** 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 $\mathsf{Th}(\mathsf{SKI})$ 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:

$$\mathsf{Th}(\mathsf{SKI} + \mathsf{R})$$

sort	t	
term constructors	$S, K, I\colon$	$1 \to t$
	$R\colon$	$t \to t$
	$(- -)\colon$	$t^2 \to t$
structural congruence	$R(x\ y)$	$= (Rx\ y)$
	$(x\ Ry)$	$= (x\ y)$
	RR	$= R$
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 $f_R\colon \mathsf{Th}(\mathsf{SKI}) \to \mathsf{Th}(\mathsf{SKI} + \mathsf{R})$:

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

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

which simply send each application to its postcomposition with R, and each rewrite to its "marked" correspondent.

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

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

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 to make do. 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. **Bisimulation.** This paper uses simple functors to illustrate the basic idea of changing semantics. Of course, there are many interesting and useful change-of-base functors. As demonstrated, any functor $F \colon \mathsf{V} \to \mathsf{W}$ which preserves finite products and finite coproducts of the terminal object can be considered as a change in semantics. For example, if we enrich in labelled directed graphs, we can utilize the important concept of *bisimulation*.

A labelled transition system consists of a set G, a label alphabet A, and a rewrite relation $\to \subset G \times A \times G$, equivalently a graph labelled by elements of A. The elements of G represent terms or processes, and the elements of A represent rewrite rules, in order to actually keep track of which kinds of rewrites are being used in a computation. An element (p, a, q) is denoted $p \xrightarrow{a} q$.

In particular, labelled transition systems allow for the correct definition of process equivalence. A **bisimilarity relation** $\equiv \subset G \times G$ consists of pairs of processes (p, q), written $p \equiv q$, defined:

$$\begin{array}{l} \forall a \in A, \;\; p', q' \in G \\ (p \xrightarrow{a} p') \text{ implies } (\exists q' \in G \;\; (q \xrightarrow{a} q') \;\wedge\; p' \equiv q') \\ (q \xrightarrow{a} q') \text{ implies } (\exists p' \in G \;\; (p \xrightarrow{a} p') \;\wedge\; p' \equiv q') \end{array}$$

Intuitively, this means that the processes p and q can always "match each other's moves" as they evolve. Then for all intents and purposes, these processes behave the same way, and hence should be considered as operationally equivalent. The **bisimulation** on G is the largest bisimilarity relation which is also a *congruence*, meaning that processes are bisimilar iff they are so in every context, i.e. when substituted into any one-hole term.

This concept, as well as the Calculus of Communicating Processes, were invented and demonstrated by Milner [26]. The latter can be expressed as an LTS-theory. The category of labelled transition systems is just like Gph, except of course we now keep track of labels. Morphisms in LTS, operations in LTS-theories, and LTS-functors all preserve labels; for example, when we compose and multiply rewrite rules, we retain this information by labelling with the actual denotation for that composite/product. Modulo these details, $\mathsf{V} = \mathsf{LTS}$ is exactly like the cases considered above.

$\mathsf{Th}(\mathsf{CCS})$

sorts	P		processes
	N		actions
	$\overline{N}$		coactions
operations	$0\colon$	$1 \to P$	nullity
	$\tau\colon$	$1 \to P$	internal action
	$\vert\colon$	$P^2 \to P$	parallel
	$+\colon$	$P^2 \to P$	choice
	$.\colon$	$N \times P \to P$	input
	$\bar{.}\colon$	$\overline{N} \times P \to P$	output
congruence		$(P, \vert, 0)$	commutative monoid
		$(P, +, 0)$	commutative monoid
rewrites	$tau\colon$	$. \circ (\tau \times P) \circ l^{-1} \Rightarrow id_P$	$(\tau.P \xrightarrow{tau} P)$
	$inter\colon$	$\vert \circ (. \times \bar{.}) \Rightarrow \vert$	$(a.P \vert \overline{a}.Q \xrightarrow{react} P \vert Q)$

The theory is summarized in the two rewrite rules: *tau* is an "unobservable" action, a process evolving in a way that is private to the ambient context; *inter* is interaction or communication - the action a is being triggered by the coaction $\overline{a}$, they are used up and the sequential processes continue in parallel. This calculus is the precursor to the π calculus [25], and is a very simple and general framework for understanding systems of interacting automata.

There is an endofunctor $B\colon \mathsf{LTS} \to \mathsf{LTS}$ which quotients by the bisimulation relation. It preserves products, $B(G \times H) \cong B(G) \times B(H)$, because $(p_1, p_2) \equiv (q_1, q_2)$ iff ($p_1 \equiv q_1$ and $p_2 \equiv q_2$). Thus we can utilize base change to perform a very useful tranformation on our semantics: from $\mathsf{Th}(\mathsf{CCS})$, we get a new theory $B_*(\mathsf{Th}(\mathsf{CCS}))$, the hom-LTS's of which consist of bisimulation equivalence classes of terms and rewrites in the calculus of communicating systems.

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-context, using an iterated Grothendieck construction can be assimilated into one category Thy, which consists of all enriched Lawvere theories. We have demonstrated these concepts with the theory of combinatory logic, $\mathsf{Th}(\mathsf{SKI})$, giving change-of-semantics from small-step to big-step to full-step to denotational semantics. Finally, we suggest that there are many interesting change-of-semantics functors, by considering an endofunctor on the category of labelled transition systems, which quotients by the bisimulation relation and is indeed a change-of-semantics.

References

[1] J. Adámek and J. Rosický, *Locally Presentable and Accessible Categories*, Cambridge U. Press, Cambridge, 1994. (Referred to on page 5, 7.)

[2] J. Baez, Can 1+1 have more than two points?, *The n-Category Café*, April 19, 2019. Available at https://golem.ph.utexas.edu/category/2019/04/can_11_have_more_than_two_poin.html. (Referred to on page 3, 14.)

[3] 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 20.)

[4] H.P. Barendregt, The Lambda Calculus, its syntax and semantics, in *Studies in Logic and The Foundations of Mathematics*, Elsevier, London, 1984. (Referred to on page 23.)

[5] 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.)

[6] J. Beardsley and L. Z. Wong, The enriched Grothendieck construction. Available as arXiv:1804.03829. (Referred to on page 19.)

[7] 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 .)

[8] R. Blackwell, G. M. Kelly and A. J. Power, Two-dimensional monad theory, *J. Pure Appl. Algebra* **59** (1989), 1–41. (Referred to on page 12.)

[9] F. Borceux, *Handbook of Categorical Algebra*, vol. 2, Cambridge U. Press, Cambridge, 1994. (Referred to on page 16, 19.)

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

[11] B. Day and R. Street, Monoidal categories and Hopf algebroids, *Adv. Math.* **127** (1997), 99–157. (Referred to on page 10, 13.)

[12] E. J. Dubuc, *Kan Extensions in Enriched Category Theory*, Lecture Notes in Mathematics, Springer, Berlin, 1970. (Referred to on page 12.)

[13] M. Fiore, *Abstract Syntax with Variable Binding*, Logic in Computer Science, IEEE Computer Science Press, 1999. (Referred to on page 20.)

[14] 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.)

[15] B. Jacobs, *Categorical Logic and Type Theory*, Elsevier, Amderstam, 1999. (Referred to on page 19.)

[16] 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, 6.)

[17] Y. Kinoshita, J. Power and M. Takeyama, Sketches, *J. Pure Appl. Algebra* **143** (1999), 275–291. (Referred to on page 8.)

[18] 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.)

[19] 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.)

[20] 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 3, 8, 9.)

[21] 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/čxl/papers/ctcs97l.pdf. (Referred to on page 2.)

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

[23] 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 .)

[24] 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.)

[25] R. Milner, The polyadic π-calculus: a tutorial, in *Logic and Algebra of Specification*, Springer, Berlin, 1993, 203–246. (Referred to on page 26.)

[26] R. Milner, Communicating and Mobile Systems: The Pi Calculus, in *Cambridge University Press*, Cambridge, UK, 1999. (Referred to on page 25.)

[27] K. Nishizawa and J. Power, Lawvere theories enriched over a general base, *J. Pure Appl. Algebra* **213** (2009), 377–386. (Referred to on page 8.)

[28] 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.)
[29] M. Hyland and J. Power, Discrete Lawvere theories and computational effects, in *Theoretical Comp. Sci.* **366** (2006), 144–162. Available at . (Referred to on page 2, 20.)
[30] J. Power, Enriched Lawvere theories, *Theory Appl. Categ.* **6**(1999), 83–93. (Referred to on page 8.)
[31] 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 20.)
[32] 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.)
[33] P. Selinger, Lecture notes on the lambda calculus. Available as arXiv:0804.3434. (Referred to on page .)
[34] 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 20.)
[35] M. Stay and L. G. Meredith, Representing operational semantics with enriched Lawvere theories. Available as arXiv:1704.03080. (Referred to on page 3, 20.)
[36] M. Stay and L. G. Meredith, Logic as a distributive law. Available as arXiv:1610.02247. (Referred to on page 20.)