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

April 21, 2019

Abstract. Enriched Lawvere theories are a generalization of Lawvere theories that allow us to describe operational semantics for 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 going 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 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. Lawvere theories [15] 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 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 [24]. 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 [27], and poset-enriched ones by Ghani and Lüth [18]. 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:

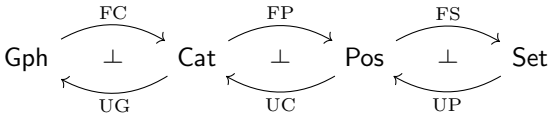
- **Graphs**: Gph-theories represent "small-step" operational semantics
  — a hom-graph edge represents a *single term* rewrite.
- **Categories**: Cat-theories represent "big-step" operational semantics*
  — composition generates morphisms representing *big-step* rewrites
  (*Conventionally, a big-step rewrite is one resulting in a "normal form", i.e. an expression which has been computed completely. We will use the term more generally.)
- **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 [17], 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: $\mathsf{Gph} \to \mathsf{Cat}$ 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" [30]. 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.

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 $\mathsf{Th}(\mathsf{Mon})$, 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 [4, Chap. 4]; for a more computer-science-oriented approach see Crole [9, 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 [10]. In 1965, Linton [16] 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 set of all possible 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:

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

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 [4, 15, 21].

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].

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 [12] 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 is compatible with 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 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 theory of adjunctions and monads generalizes to the V-enriched context, and the adjunction

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

is called "currying"; its counit is "evaluation".

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

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

that are V-natural in a. If such an object 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** 1_{C}. Whenever V is cartesian closed, the finite products in V are also V-products in $\underline{\mathsf{V}}$; this 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.

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

$$C(b, a) \cong \sum_{i=1}^{n} C(b_i, a)$$

V-natural in a. 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 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.

A bit more subtly, generalizing how V has a product and internal hom, 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 $v \in V$, 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 $\mathsf{C} = \overline{V}$ this holds for all $v \in V$, 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" [25]. 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 this context.

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 [13] introduced "enriched sketches" as a way of presenting enriched Lawvere theories, and "monads with arities" have been explored by Berger, Melliés and Weber [6], but they are not yet widely understood or used. What does it really mean for an operation to take in a finite graph of terms? 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.

In this paper, however, we only consider *natural number* arities, while still retaining enrichment. To do this we use the work of Lucyshyn-Wright [17], who along with Power [23] 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 the **category of models** of (T, τ) in C, $\mathsf{Mod}(\mathsf{T}, \mathsf{C})$, to consist of V-functors $\mu \colon \mathsf{T} \to \mathsf{C}$ that preserve finite V-products and V-natural transformations between these.

Definition 4. We define the **category of V-theories**, VLaw, 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 obeys $f\tau = \tau'$.

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

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

sending any model $\mu \colon \mathsf{T} \to \mathsf{V}$ to its underlying object $\mu(t) \in \mathsf{V}$. 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 category of algebras of the resulting monad $T = UF$. More precisely:

Theorem 5. Suppose V is cartesian closed and cocomplete, with chosen coproducts of the terminal object, and 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 [17], 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 11 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 12 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 complete. In Thm. 8.9 shows that when $j\colon \mathsf{J} \to \underline{\mathsf{V}}$ is eleutheric, we may form the V-category $\underline{\mathsf{Mod}}(\mathsf{T},\mathsf{V})$, and that the forgetful V-functor

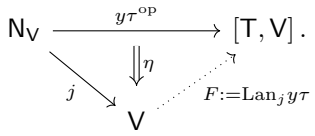
$$U\colon \underline{\mathsf{Mod}}(\mathsf{T},\mathsf{V}) \to \underline{\mathsf{V}}$$

is monadic. This is the result we need. □

Before proving the required lemmas, 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 $\tau^{\mathrm{op}}\colon \mathsf{A}_{\mathsf{V}} \to \mathsf{T}^{\mathrm{op}}$ with the enriched Yoneda embedding $y\colon \mathsf{T}^{\mathrm{op}} \to [\mathsf{T},\mathsf{V}]$:

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

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

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

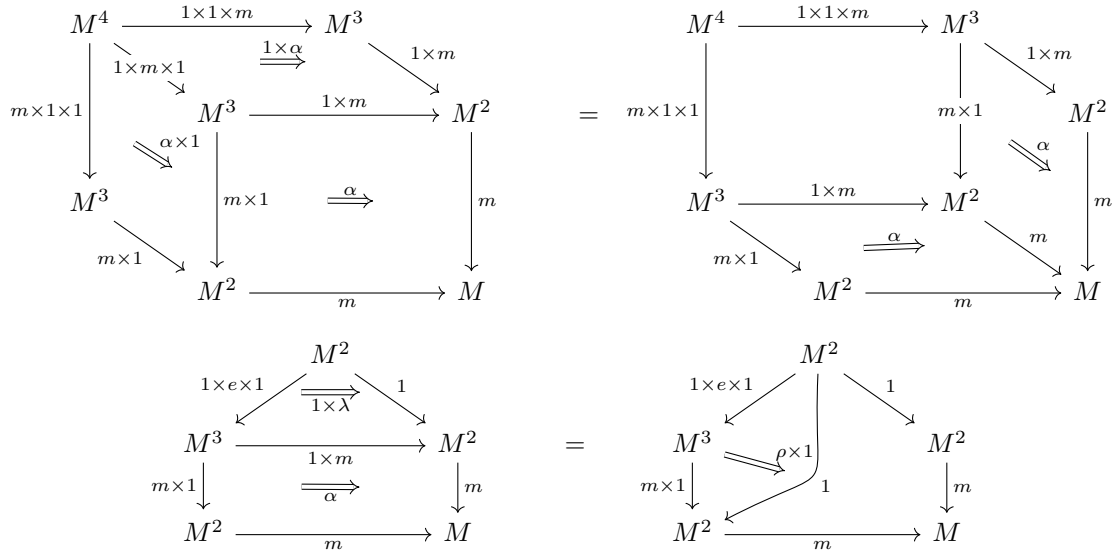
and this has a more concrete formula as an enriched coend:

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

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 [14]. 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:

Th(PsMon)

sorts	M	pseudomonoid
operations	$e\colon 1 \to M$	identity
	$m\colon M^2 \to M$	multiplication
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
coherence laws		



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. **CHECK - SHOULDN'T THERE BE TWO UNITOR LAWS???**

Models of $\mathsf{T} = \mathsf{Th}(\mathrm{PsMon})$ in Cat are monoidal categories. A Cat-enriched monad is called a 2-monad, and T gives rise to the "free monoidal category" 2-monad on Cat [7]. 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

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

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 operations obeying equations to equipping *category*-like objects with operations obeying equations up to transformations that obey certain equations of their own.

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) 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". Finally, an algebra of the monad

$$T = UF \colon \mathsf{C} \mapsto \int^n \mathsf{C}^n \times \mathsf{T}(n, 1)$$

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, $(A, \otimes_A, I_A)$ is precisely a monoidal category, and we have the equivalence:

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

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 7. 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

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

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 8. 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 7 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 $|\hom(m_\mathsf{V}, n_\mathsf{V})| = n^{km}$. Richard Garner has shown us a simple argument showing that $|\hom(1_\mathsf{V}, 2_\mathsf{V})|$ must be a power of 2 if it is finite. Namely, 2_V is an internal Boolean algebra in V, so its set of points must be a Boolean algebra in Set, and thus its cardinality must be a power of 2 when finite. It should be possible to completely understand the structure of N_V as a category, but this would be a digression from our current mission.

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 9. 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_V} exists, we have

$$\begin{array}{rcll} \mathsf{C}(a, c^{n_\mathsf{V}}) & \cong & \mathsf{C}(a,c)^{n_\mathsf{V}} & \text{definition of powers} \\ & \cong & \mathsf{C}(a,c)^n & \text{Lemma 7} \end{array}$$

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

$$\begin{array}{rcll} \mathsf{C}(a, c^n) & \cong & \mathsf{C}(a,c)^n & \text{definition of } \mathsf{V}\text{-products} \\ & \cong & \mathsf{C}(a,c)^{n_\mathsf{V}} & \text{Lemma 7} \end{array}$$

and each of these isomorphisms is V-natural in a, so c^n is the power of c by n_V. $\square$

Lemma 10. 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

$$\begin{array}{rcll} F(b^{n_\mathsf{V}}) & \cong & F(b^n) & \text{Lemma 9} \\ & \cong & F(b)^n & \text{preservation of products} \\ & \cong & F(b)^{n_\mathsf{V}} & \text{Lemma 9} \end{array}$$

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

$$\begin{array}{rcll} F(1_\mathsf{C}) & \cong & F(c^0) & \mathsf{V}\text{-terminal object is a 0-ary product} \\ & \cong & F(c^{0_\mathsf{V}}) & \text{Lemma 9} \\ & \cong & F(a)^{0_\mathsf{V}} & \text{preservation of } \mathsf{N}_\mathsf{V}\text{-powers} \\ & \cong & F(a)^0 & \text{Lemma 9} \\ & \cong & 1_\mathsf{D} & \mathsf{V}\text{-terminal object is a 0-ary product.} \end{array}$$

For the latter note that

$$\begin{array}{rcll} F(c^m \times c^n) & \cong & F(c^{m+n}) & \mathsf{V}\text{-products in } \mathsf{C} \\ & \cong & F(c^{(m+n)_\mathsf{V}}) & \text{Lemma 9} \\ & \cong & F(c)^{(m+n)_\mathsf{V}} & \text{preservation of } \mathsf{N}_\mathsf{V}\text{-powers} \\ & \cong & F(c)^{m+n} & \text{Lemma 9} \\ & \cong & F(c)^m \times F(c)^n & \text{terminal object is a 0-ary product.} \end{array}$$

□

THE ABOVE ARGUMENT IS SLOPPY since "preserving V-products" probably means more than just $F(a \times b) \cong F(a) \times F(b)$, since it means more than that for "preserving products". Maybe something similar is true for preserving N_V-powers. WE NEVER DEFINE THE CONCEPTS!!!

Lemma 11. 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 10 implies that conditions 2 and 3 are equivalent. □

Lemma 12. 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 10 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 $F_*\colon \mathsf{VCat} \to \mathsf{WCat}$ [8] 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 $F(f)$.

We can also define F_* on enriched natural transformations. 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

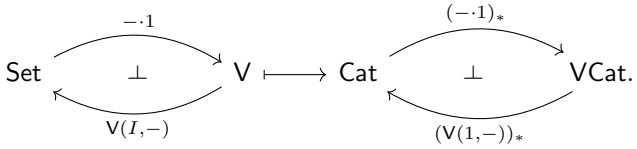
$$1 \xrightarrow{\sim} F(1) \xrightarrow{F(i_a)} F(\mathsf{C}(a,b)).$$

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. In Theorem 14 we see that everything works as one would hope.

We want to know when the functor $F\colon \mathsf{V} \to \mathsf{W}$ induces a change of base $F_*\colon \mathsf{VCat} \to \mathsf{WCat}$ 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 are requiring that V and W be cartesian closed, equipped with chosen finite coproducts of their terminal object. 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 8 in these computation.

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.

Lemma 13. Let $F\colon \mathsf{V} \to \mathsf{W}$ be a cartesian functor that preserves chosen finite coproducts of the terminal object. If $f\colon \mathsf{C} \to \mathsf{D}$ is an N_V-power-preserving V-functor, then $F_*(f)\colon F_*(\mathsf{C}) \to F_*(\mathsf{D})$ is an N_V-power-preserving W-functor.

Proof.

$$\begin{array}{rcll}
F_*(\mathsf{D})(F_*(f)(a), F_*(f)(s^{n_\mathsf{V}})) & = & F(\mathsf{D}(f(a), f(s^{n_\mathsf{V}})) & \text{definition of base change} \\
& \cong & F(\mathsf{D}(f(a), f(s)^{n_\mathsf{V}}) & f \text{ preserves } \mathsf{N}_\mathsf{V}\text{-powers} \\
& \cong & F(\mathsf{D}(f(a), f(s))^n) & \text{Lemma 7 for } \mathsf{V} \\
& \cong & F(\mathsf{D}(f(a), f(s)))^n & F \text{ cartesian} \\
& = & F_*(\mathsf{D})(f(a), f(s))^n & \text{definition of base change} \\
& \cong & F_*(\mathsf{D})(f(a), f(s)^{n_\mathsf{W}}) & \text{Lemma 7 for } \mathsf{W}. \quad \square
\end{array}$$

Finally, let $\tilde{n}\colon \underline{\mathsf{N}}_\mathsf{W} \to F_*(\underline{\mathsf{N}}_\mathsf{V})$ 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 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 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. We have seen that τ_{W} is a W-theory. The preservation of models follows from Lemma 13. □

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 $(\mathsf{N}_{\mathsf{V}})^n$ is also an eleutheric system of arities. In Section 8.4, 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. **NO, we need preservation of chosen finite coproducts of the terminal object as well!!! Fix this everywhere!!! We need a different category, not CCC, below!!!**

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 [8, 11]. We noted in §4 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 [5]; 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 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 $\mathrm{Thy} := \int \mathrm{thy}$ 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* [31]. 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*:

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

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

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** [26], 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 [29].

The key insight of Stay and Meredith [30] 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:	$1 \to t$
	K:	$1 \to t$
	I:	$1 \to t$
	$(- -)$:	$t^2 \to t$
structural congruence	n/a	
rewrites	σ:	$(((S\ -)\ =)\ \equiv) \Rightarrow ((-\ \equiv)\ (= \equiv))$
	κ:	$((K\ -)\ =) \Rightarrow -$
	ι:	$(I\ -) \Rightarrow -$

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:

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

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

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

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

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

(Morphisms $1 \to t$ are the "closed" terms, meaning they have no 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):

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

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,

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

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

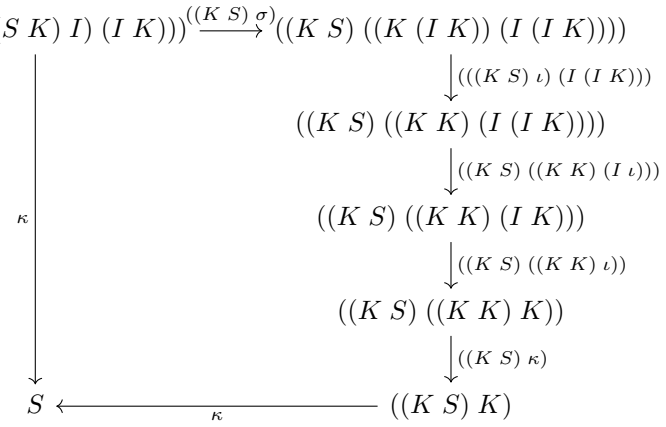
These rewrites in the image of μ are *graph transformations*, which are just like natural transformations of functors, without the commuting diagram: given two graph homomorphisms $f, g\colon G \to H$,

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 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:



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:

$$\mathsf{Gph} \underset{u_{\mathsf{Gph}}}{\overset{f_{\mathsf{Gph}}}{\rightleftarrows}} \mathsf{Mod}(\mathsf{Th}(SKI), \mathsf{Gph}) \quad (\perp)$$

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

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

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 $\mathrm{FC}\colon \mathsf{Gph} \to \mathsf{Cat}$ 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

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

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

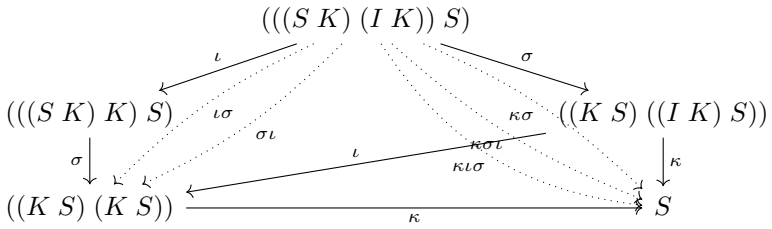
is the category which represents the big-step operational semantics of the SKI-calculus. To correpond with the conventional meaning of big-step semantics, as noted in §1, we can quotient by Curry's equations to identify SKI-terms which are extensionally equivalent [3].

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. 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 $\mathrm{FS}\colon \mathsf{Pos} \to \mathsf{Set}$ of the functor $\mathrm{UP}\colon \mathsf{Set} \to \mathsf{Pos}$ 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 a pair of connected components corresponds to a connected component of pairs).

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. (One more note here - the untyped SKI calculus is actually not terminating, and so we need to consider the simply typed SKI-calculus [3].)

Modulo a couple technicalities, 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:



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:	$1 \to t$
	R:	$t \to t$
	$(- -)$:	$t^2 \to t$
structural congruence	$R(x\ y) = (Rx\ y)$	
rewrites	σ_r:	$(((RS -) =) \equiv) \Rightarrow ((R- \equiv) (= \equiv))$
	κ_r:	$((RK -) =) \Rightarrow R-$
	ι_r:	$(RI -) \Rightarrow R-$

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 [22], 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" [20].

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 [30] §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 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 18.)

[3] 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 21.)

[4] 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.)

[5] J. Beardsley and L. Z. Wong, The enriched Grothendieck construction. Available as arXiv:1804.03829. (Referred to on page 16.)

[6] 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.)

[7] 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 10.)

[8] F. Borceux, *Handbook of Categorical Algebra*, vol. 2, Cambridge U. Press, Cambridge, 1994. (Referred to on page 14, 16.)

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

[10] 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.)

[11] B. Jacobs, *Categorical Logic and Type Theory*, Elsevier, Amderstam, 1999. (Referred to on page 16.)

[12] 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.)
[13] Y. Kinoshita, J. Power and M. Takeyama, Sketches, *J. Pure Appl. Algebra* **143** (1999), 275–291. (Referred to on page 7.)
[14] B. Day and R. Street, Monoidal categories and Hopf algebroids, *Adv. Math.* **127** (1997), 99–157. (Referred to on page 9.)
[15] 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.)
[16] 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.)
[17] 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, 7, 8.)
[18] 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.)
[19] S. Mac Lane, *Categories for the Working Mathematician*, Springer, Berlin, 1998. (Referred to on page .)
[20] 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 23.)
[21] 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.)
[22] R. Milner, The polyadic π-calculus: a tutorial, in *Logic and Algebra of Specification*, Springer, Berlin, 1993, 203–246. (Referred to on page 23.)
[23] K. Nishizawa and J. Power, Lawvere theories enriched over a general base, *J. Pure Appl. Algebra* **213** (2009), 377–386. (Referred to on page 7.)
[24] 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.)
[25] J. Power, Enriched Lawvere theories, *Theory Appl. Categ.* **6**(1999), 83–93. (Referred to on page 7.)
[26] 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 18.)
[27] 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.)
[28] P. Selinger, Lecture notes on the lambda calculus. Available as arXiv:0804.3434. (Referred to on page .)
[29] 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 18.)
[30] M. Stay and L. G. Meredith, Representing operational semantics with enriched Lawvere theories. Available as arXiv:1704.03080. (Referred to on page 3, 18, 23.)
[31] M. Stay and L. G. Meredith, Logic as a distributive law. Available as arXiv:1610.02247. (Referred to on page 17.)