慣性ドリフト: From 0 to Normalization by Gluing in 4.9 seconds
A Brisk Drift through Categorical Semantics of Lambda Calculi

---

A Thesis
Presented to
The Established Interdisciplinary Committee for Mathematics and Computer Science Mathematics and Natural Sciences
Reed College

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Arts

Jay Kruer

December 2021

Approved for the Division
(Mathematics)

Angélica Osorno	James (Jim) Fix

Acknowledgements

Preface

> And further, by these, my son, be admonished: of making many books there is no end; and much study is a weariness of the flesh.
>
> Ecclesiastes 12:12, KJV.

I don't know what to put here, but this is certainly a funny quote that should find a home somewhere in the thesis.

List of Abbreviations

TMA Too Many Abbreviations

Table of Contents

List of Tables

List of Figures

Abstract

Normalization by gluing is poggers.

Introduction

Blah blah blah, I should write some stuff about how category theory helps us avoid insane proof heuristics in metatheory of type theory.

Chapter 1

Functorial semantics of type theory: sketches and their models; algebraic theories and their algebras

In this chapter we develop tools for reasoning about (syntactic) *theories*, which are in some sense "notions of" abstract structure. Examples of theories include the theory of rings, and simple type theory. To discuss how we write down our theories, we need another level of abstraction. We will work with several *notions of* (syntactic) theory, and we will more laconically refer to a notion of syntactic theory as a *doctrine*. A *doctrine* is something like a meta-framework specifying how we are to write down a theory. The first doctrine we will consider is that of the *elementary sketch*. The doctrine of the *elementary sketch* allows us to write down theories involving *unary* operations (those defined over a single argument.) This restriction on the arity of operations turns out to be quite limiting. To address this, we will later upgrade the doctrine of elementary sketches to the doctrine of *algebraic theories* which allow encoding operations with any finite number of arguments, thus covering a broad variety of theories. Algebraic theories are known more famously as *Lawvere theories* after categorical logic superstar (and former Reed College professor!) William Lawvere who originally studied them while building his functorial treatment of universal algebra. Keeping with our sketchy terminology and emphasizing the doctrinal upgrade, algebraic theories are also called *finite product sketches*. As an example of the strength of algebraic theories, we will show how to write down (as an algebraic theory) what it means to be a ring, with no reference to sets or functions. In the following chapter, we will use the doctrine of algebraic theories to develop categorical semantics of the lambda calculus. This chapter is strictly expository in nature. Much of the following presentation draws heavily from Paul Taylor's *Practical Foundations of Mathematics* [**taylor_practical_1999**]. Our humble contribution is to flesh out some of his examples, add some of our own, and make parts of the presentation more palatable and quickly digestible to the reader already acquainted with basic category theory and type theory.

1.1 Elementary sketches and their models

1.1.1 An algebraic prelude

We begin by recalling from algebra the notion of an *action* of, say, a group or a monoid. Actions are, from our perspective, a way of giving meaning, or *semantics* to elements of a set which enjoys some algebraic structure.

Definition 1. Recall that a **covariant action** of a group or monoid $(M, id, \cdot)$ on a set A is a binary operation $(-)_* (=) : M \times A \to A$ such that $\mathrm{id}_* a = a$ and $(g \circ f)_* a = g_* (f_* a)$. We can similarly a define the notion of a **contravariant action** which similarly requires the identity action to do nothing, but instead flips the order of action for compositions: $(g \circ f)^* a = f^* (g^* a)$

For example, in algebra we learn that the dihedral group of order 8, written D_4, acts on the square (with uniquely identified points) by reflections and rotations[1]; each of the operations encoded by the action results in the same image of the square up to ignoring the unique identity of the points we started with. This action gives geometric meaning to each of the group elements, and was used in the first day of the author's algebra class to explain the algebraic mechanics of the group itself; discovering which elements of the group are inverse to one another is done by geometric experimentation using a square with uniquely colored vertices. Similarly, the symmetric groups S_n act on lists of length n by permutation of the list elements. In this case, the action can be even more trivially defined. We now turn to the definition of an important property of actions: *faithfulness*.

Definition 2. A **faithful** action $(-)_*$ is one for which things are *semantically* equal (or, act the same) only when they are *syntactically* equal (or, *are* the same as far as your eyeballs are concerned.) More precisely rendered, we require:

$$\forall (a : A) . f_* a = g_* a \implies f = g$$

It can now be seen that the crucial property enjoyed by the natural action of D_8 on the square which enabled our use of paper cutouts in studying the group is faithfulness. If the action were not faithful, determining which operations in D_8 are inverses would not be so easy as printing out a square and plugging away, because we may (among other catastrophes) be working with an action which may not take only the identity element to the leave-everything-in-place operation on the square.

Having gesticulated that actions gives groups and monoids their meaning, we turn to the development of the doctrines of *elementary sketch* and *algebraic theory* which will allow us to generalize both sets with algebraic structures and their actions to new settings.

1.1.2 Elementary sketches

As promised, we begin with the definition.

[1] https://groupprops.subwiki.org/wiki/Dihedral_group:D8

Definition 3. An **elementary sketch** is comprised of the following data:

1. a collection $X, Y, Z, ...$ of named **base types** or **sorts**
2. a **variable** $x : X$ for each occurrence of each named sort.
3. a collection of **unary operation-symbols** or **constructors** τ having at most one variable. As a clarifying example: when sketching type theories, we will write $x : X \vdash \tau(x) : Y$.
4. a collection of equations or **laws** of the form:

$$\tau_n \left(\tau_{n-1} \left(\cdots \tau_2 \left(\tau_1 \left(x\right)\right) \cdots\right)\right) = \sigma_m \left(\sigma_{m-1} \left(\cdots \sigma_2 \left(\sigma_1 \left(x\right)\right) \cdots\right)\right)$$

We will discuss the generality provided by this definition after some intervening examples. One of the simplest examples is the sketch of a (free) monoid on some set S:

Example 1 (Sketch of a (free) monoid)**.** The requisite data for the sketch are as follows:

1. The collection of sorts is the singleton $\{M\}$.
2. The collection of variables is $\{m : M\}$.
3. The collection of operation symbols is the set S. Each has as its signature $M \to M$
4. No equations are imposed.

To really buy that this sketch generates a free monoid, we need an intervening definition of a concept we will get a lot of mileage out of in this thesis. The idea should already be familiar from our study of type theory, despite the drastically simplified setting.

Definition 4. Given an elementary sketch, a **term** $x : \Gamma \vdash X$ is a string of composable unary operation-symbols applied to a variable $\gamma : \Gamma$ as in: $\tau_n \left(\tau_{n-1} \left(\cdots \left(\tau_2 \left(\tau_1 \left(\gamma\right)\right)\right)\right)\right)$. Composable unary-operation symbols are ones which have compatible domains and codomains in the usual sense as in set theory.

We now propose a more precise version of the above claim: the terms of the sketch defined above form the elements of a free monoid over S. Before we can continue, we should decide what our term composition will be.

Definition 5 (Composition of terms)**.** Composition of terms is by substitution for the variable: for a term $\sigma : \Delta \vdash \Gamma$ and some terms τ_i with $\tau_n : \cdots \vdash \Xi$ we define

$$\left(\tau_n \left(\tau_{n-1} \left(\cdots \left(\tau_2 \left(\tau_1 \left(\gamma\right)\right)\right)\right)\right)\right) \circ \sigma = \tau_n \left(\tau_{n-1} \left(\cdots \left(\tau_2 \left(\tau_1 \left(\sigma \left(\delta\right)\right)\right)\right)\right)\right) : \Delta \vdash \Xi.$$

With our notion of composition in hand, we can now handwave an argument for our revised claim that the terms of the sketch form the elements of a free monoid. The reader will recall from our early discussions of basic type theory that substitution is associative [TODO in Ch 1]. As a consequence, any elementary sketch, including this one, satisfies that axiom for free. The identity term is given by zero composable unary operation-symbols applied to a variable. It's just a variable; when composing the identity term with any other term, we end up getting exactly that original term.

This loose argument is somewhat satisfying, but we can do better. To get there, we will first develop a notion generalizing *actions* from algebra. After doing so, we will give more concrete meaning to this sketch and complete our intuitive handle on it.

Definition 6. A **model** (also known as an algebra, an interpretation, a covariant action) of an elementary sketch is comprised of:

1. an assignment of a set A_X to each sort X and
2. an assignment of a function $\tau_* : A_X \to A_Y$ for each operation-symbol of the appropriate arity such that:
3. each law is preserved; i.e., for each law as before we have

$$\tau_{n_*}\left(\tau_{n-1_*}\left(\cdots\tau_{2_*}\left(\tau_{1_*}(x)\right)\cdots\right)\right) = \sigma_{m_*}\left(\sigma_{m-1_*}\left(\cdots\sigma_{2_*}\left(\sigma_{1_*}(x)\right)\cdots\right)\right)$$

that is, the covariant action on operation-symbols is faithful in the sense defined above.

The next definition will feature prominently in our later study of type theory, but will prove useful in studying Example 1 by forming the sets of a "for-free" model for any elementary sketch.

Definition 7. Given an elementary (unary) sketch, the **clone** at (Γ, X) is the set $\mathrm{Cn}_{\mathcal{L}}(\Gamma, X)$ of all the **terms** of sort X assuming a single variable of sort Γ, quotiented by the laws of the sketch.

The fact that a sketch's clones contain *equivalence classes* (with respect to the laws) of its terms will feature prominently in our later study of ideas central to the goals of this thesis. In particular, clones alone don't allow for any meaningful discussion of computational behavior of terms undergoing reduction; a term's normal form and its various reducible forms are identified in the clone.

It can be shown that the clones of a sketch form (the sets for) a model of a sketch. In particular, it can be shown that the sketch acts covariantly on the set of its clones:

Theorem 1. *Every elementary sketch has a faithful covariant action on its clones* $\mathcal{H}_X = \cup_\Gamma Cn_{\mathcal{L}}(\Gamma, X)$ *by sequencing with the operation symbol. Substitution for the (single) variable in a term gives a faithful contravariant action on* $\mathcal{H}^Y = \cup_\Theta Cn_{\mathcal{L}}(Y, \Theta)$.

Proof. The actions of $\tau : X \to Y$ on $\mathrm{Cn}_{\mathcal{L}}(\Gamma, X) \subseteq \mathcal{H}_X$ and $\mathrm{Cn}_{\mathcal{L}}(Y, \Theta) \subseteq \mathcal{H}^Y$ are given by:

- $\tau_* a_n \left(\cdots a_2 \left(a_2 \left(\sigma\right)\right) \cdots\right) = \tau\Big(a_n \Big(\cdots \big(a_2 \left(a_1 \left(\sigma\right)\right)\big)\Big)\Big) \in \mathrm{Cn}_{\mathcal{L}} \left(\Gamma, Y\right)$
- $\tau^* \zeta_m \left(\cdots \zeta_2 \left(\zeta_1 \left(y\right)\right)\right) = \zeta_m \Big(\cdots \zeta_2 \left(\zeta_1 \left(\tau(x)\right)\right)\Big) \in \mathrm{Cn}_{\mathcal{L}} \left(X, \Theta\right)$

where $\sigma : \Gamma, x : X,$ and $, y : Y$. Covariance of the former is clear. Contravariance of the latter follows by considering the behavior of substitutions in sequence. □

Recalling our sketch of a monoid from Example 1, the substance of this covariant action morally amounts to saying that the sketch acts on its terms by left multiplication (here "multiplication" is actually just juxtaposition plus some parentheses) which gives the robust version of the handwavy argument we provided above.

Joke 1. A couple of type theorists walk into a Michelin starred restaurant. The menu reads in blackboard bold letters "ℕ𝕆 𝕊𝕌𝔹𝕊𝕋𝕀𝕋𝕌𝕋𝕀𝕆ℕ𝕊". They promptly leave.

1.1.3 The category of contexts and substitutions

We now introduce a very special category. This category is special in both the structure it enjoys as well as the central role it will play in the rest of the thesis. This category goes by many names: *syntactic category*, the (rather verbose) *category of contexts and substitutions*, and the elusive *classifying category*. We endeavor to explain the meaning behind each of these names over the course of the thesis, but for now we adopt the name most closely describing its presentation.

Definition 8 (The category of contexts and substitutions)**.** Given a sketch $\mathcal{L}$, the **category of contexts and substitutions**, written $\mathrm{Cn}^{\times}_{\mathcal{L}}$ is presented as follows:

- The objects are the contexts of $\mathcal{L}$, i.e., finite lists of distinct variables and their types.
- The generating morphisms are:
  - Single substitutions or *declarations* $\left[a/x\right] : \Gamma \to \left[\Gamma, x : X\right]$ for each term $\Gamma \vdash a : X$. The direction in the signature should be confusing unless you're either already an expert or a total novice to type theory.
  - Single omissions or *drops* $\hat{x} : \left[\Gamma, x : X\right] \to \Gamma$ for each variable $x : X$.
- The laws are given by an extended version of the familiar substitution lemma from type theory. The following laws are added for each collection of terms a, b and distinct variables x and y such that x does not appear free in a and y appears free in neither a or b:

$$\begin{aligned} \left[a/x\right] ; \hat{x} &= \mathrm{id} \\ \left[a/x\right] ; \left[b/y\right] &= \Big[\left[a/x\right]^* b/y\Big] ; \left[a/x\right] \\ \left[a/x\right] ; \hat{y} &= \hat{y}; \left[a/x\right] \\ \hat{x}; \hat{y} &= \hat{y}; \hat{x} \\ \left[x/y\right] ; \hat{x}; \left[y/x\right] ; \hat{y} &= \mathrm{id} \end{aligned}$$

We will briefly speak to the meaning of the laws. The first law says that binding a variable to some term and then forgetting the variable is just the same as doing nothing. The second law says that successive variable declarations commute *up to accounting for the first declaration in the body of the second*. The third law says that *non-overlapping* declarations and drops commute. The fourth law says that pairs of non-overlapping drops commute. The last law is tricky and is easier to explain by passing to the substitution point-of-view. Since the substitution functor is contravariant, this requires considering the compositions in reverse order as:

$$\widehat{y}^*; [y/x]^* ; \widehat{x}^*; [x/y]^* = \mathrm{id}^*$$

Rendered thus, this law means that introducing a free variable y to the context[2], followed by replacing every free occurrence of x with y, followed by re-introducing x as a variable in the context, and then finally replacing every free occurrence of y with x is the same as doing nothing. More concisely at the expense of precision, renaming a free variable in a term and then un-renaming it results in the same term.

This category, as with most in category theory, serves to allow us to define a special class of functor. In our case, that class of functor captures what it means to produce a model of an elementary sketch. The proof of this theorem is rather bureaucratic, but its importance is that it teaches us that the canonical elementary language of a category is purpose-built so that its models are precisely set-valued functors out of the category in question.

1.1.4 Models are essentially set-valued functors on the category of a sketch

Theorem 2 (The classifying category)**.** *Let $\mathcal{L}$ be an elementary sketch and $\mathrm{Cn}_{\mathcal{L}}$ the category it presents. Then the models of $\mathcal{L}$ correspond to functors $\mathrm{Cn}_{\mathcal{L}} \to \mathfrak{Set}$.*

Proof. ($\Rightarrow$) Suppose we have an $\mathcal{L}$-model A. Then A is an assignment of a set $\lceil X \rceil_A$ to each sort $\lceil X \rceil$ and an assignment of a function $\lceil r \rceil_A : X_A \to Y_A$ to each operation-symbol $X \vdash \lceil r \rceil (x) : Y$ such that the laws (given by Theorem **??**) of $\mathcal{L}$ are preserved. These assignment form precisely the data of a functor

$$\begin{aligned} F_A : \mathrm{Cn}_{\mathcal{L}} &\to \mathfrak{Set} \\ X &\mapsto \lceil X \rceil_A \\ \left(X \xrightarrow{r} Y\right) &\mapsto \lceil r \rceil_A \end{aligned}$$

.

It remains to show functoriality of these assignments which follow from the laws of the canonical elementary language and faithfulness of the model.

[2] possibly having no effect if y is already present

($\Leftarrow$) Suppose we have a functor $F_A : \mathrm{Cn}_{\mathcal{L}} \to \mathfrak{Set}$. We will construct a model A of $\mathcal{L}$ from F_A as follows: Recall from Theorem **??** that the sorts $\lceil X \rceil$ of the sketch $\mathcal{L}$ are precisely the objects X of $\mathrm{Cn}_{\mathcal{L}}$, and the operation symbols $X \vdash \lceil f \rceil : Y$ are the morphisms f of $\mathrm{Cn}_{\mathcal{L}}$. Now,

1. For each sort $\lceil X \rceil$ we assign $\lceil X \rceil_A = F_A(X)$.

2. For each operation symbol $\lceil f \rceil$ we assign $\lceil f \rceil_A = F_A(f)$.

3. Again by Theorem **??**, the only laws of the sketch are that $\lceil \mathrm{id} \rceil(x) = x$ and $\lceil g \rceil(\lceil f \rceil(x)) = \lceil g \circ f \rceil(x)$. According to the assignments in the previous two points, the first law says that $F_A(\mathrm{id}_X) = \mathrm{id}_{\lceil X \rceil_A}$, and the second says that $(F_A \lceil g \rceil) \circ (F_A(\lceil f \rceil)) = F_A(g \circ f)$. Both are ensured by functoriality.

□

1.2 Algebraic theories and their algebras

Having defined elementary sketches, which give us a way to define multi-sorted theories, it's obvious to request the ability to define multi-input operations[3]. Algebraic theories generalize the doctrine of elementary sketches and allow us to do so. As we upgrade our doctrine to allow products, many of the notions (*terms, clones, syntactic category, etc.*) which we developed in the simplified world of elementary sketches will come along for the ride.

Definition 9 (Algebraic theory)**.** A (finitary many-sorted) **algebraic theory** $\mathcal{L}$ has

1. a collection Σ of base types or **sorts** X

2. an inexhaustible collection of variables $x_i : X$ of each sort;

3. a collection of **operation symbols**, $X_1, \ldots, X_k \vdash r : Y$ each having an **arity**, namely a list of input sorts X_i, and an output sort Y; and

4. a collection of **laws**, posed as equalities between different terms (in the sense defined before)

The next major concept we will introduce generalizes to algebraic theories the notion of *action* or *model* we saw previously for elementary sketches. As expected, the definition will be essentially the same up to taking some products. Before doing so, we will give an intervening example of an algebraic theory.

[3]Here's a little known statistic: At least one in two readers of this draft will observe that the doctrine of algebraic theory can be rephrased in terms of operards: algebraic theories are operads for which the tensor product used in forming the operation domains happens to be the plain ol' Cartesian product [**TODO: Nlab**]

Example 2 (Algebraic theory of *ring*)**.** We sketch an algebraic theory encoding the familiar structure of a ring from abstract algebra. The presentation should look familiar (when squinting) to anyone with a background in abstract algebra, except that we force the existence of multiplicative and additive identities by requiring any model (to be defined!) of this theory to provide *global elements*, namely operations out of a distinguished sort $\mathbb{1}$.

1. Sorts: The sorts are $\mathbb{1}, S$. The variable collections for each sort are $\{\,\square\,\} \cup \{\,\square_i\,\}_i$ and $\{\,s_i\,\}_i \cup x, y, z$ respectively.

2. Operations:

$$\begin{aligned} \cdot &: S \times S \to S, \\ + &: S \times S \to S, \\ 0 &: \mathbb{1} \to S, \\ 1 &: \mathbb{1} \to S, \\ - &: S \to S \end{aligned}$$

3. Laws:

$$\begin{aligned} +(x, y) &= +(y, x) \\ +(0(\square), x) &= x \\ +(x, -(x)) &= 0(\square) \\ \cdot(x, y) &= \cdot(y, x) \\ \cdot(1(\square), x) &= x \\ \cdot(x, +(y, z)) &= +(\cdot(x, y), \cdot(x, z)) \end{aligned}$$

Having given the obligatory concrete example, we now have permission to proceed with another abstract definition: that of an $\mathcal{L}$*-algebra* for an algebraic theory:

Definition 10 ($\mathcal{L}$-algebra)**.** Given an algebraic theory $\mathcal{L}$ and a category C with finite products (in the sense of the universal property as treated in the chapter on basic category theory) an $\mathcal{L}$*-algebra in* C is comprised of

1. an object A_X of C for each sort X of $\mathcal{L}$, and

2. for each operation symbol $X_1, \ldots, X_k \vdash r : Y$, an assignment of a map $r_A : A_{X_1} \times \cdots \times A_{X_k} \to A_Y$ in C.

such that the assignments respect the laws of $\mathcal{L}$.

We are now in good shape to give an example of an algebra (in the category of sets) for the theory of a ring given in Example 2.

Example 3. 1. For the sorts, we set $A_S = \mathbb{Z}$ and $A_{\mathbb{1}} = \{\star\}$

2. For the operations, we set

   (a) $\cdot_A = *$ where $*$ is the ordinary multiplication of integers

   (b) $+_A = +$ where the second plus is ordinary addition of integers

   (c) 0_A to the constant function $x \mapsto 0 \in \mathbb{Z}$

   (d) 1_A to the constant function $x \mapsto 1 \in \mathbb{Z}$

   (e) $-_A$ to the function $x \mapsto -x$ taking an integer to its additive inverse

3. We wouldn't dare bore the reader by verifying all the laws, so we demonstrate just one. We show that the 0 selected by the model indeed serves as the left identity of addition in the model.

   Proof.

$$\begin{aligned} +_A \circ \langle 0_A, \mathrm{id} \rangle &= (x : \{\star\}, y : \mathbb{Z}) \mapsto 0_A\,(x) + \mathrm{id}\,(y) \\ &= (x : \{\star\}, y : \mathbb{Z}) \mapsto 0_{\mathbb{Z}} + y \\ &= (x : \{\star\}, y : \mathbb{Z}) \mapsto y \\ &= \mathrm{id}_{\mathbb{1}_Z \times S_A} \end{aligned}$$

   Our proof is almost done, but we must justify that the final identity morphism is actually the interpretation of the variable (regarded as a term) x. This fact will be validated by results later in this section. In particular, Definition 12 will give a proper treatment to the interpretation of terms in an algebraic theory and allow us to justify the equivalence of the terms x and $\widehat{\square}^* x$ by[4] way of our construction of the syntactic category. With the clearing of that remaining goal-post deferred, we have shown what is required for this law. □

The reader familiar with algebra will observe that this example (at least when fully worked out by a less lazy typist) amounts to verifying that the integers form a ring under the standard multiplication and addition operations we learn in elementary school. A natural next question to ask is how we might encode a ring homomorphism in this framework. To answer this question, we (of course) define a more general notion:

Definition 11 ($\mathcal{L}$-algebra homomorphism)**.** A *homomorphism* $A \to B$ of $\mathcal{L}$-algebras A and B in some category $\mathfrak{C}$ with finite-products is an assignment to each sort X of a $\mathfrak{C}$-morphism $\phi_X : A_X \to B_X$ between the corresponding objects in each algebra such that diagrams of the following form commute:

[4] recall from Definition 8 that $\square$ is the variable we settled on for the sort $\mathbb{1}$ and substitution by the hat is context weakening or adding the variable

$$\begin{array}{ccc} A_{X_1} \times \cdots \times A_{X_k} & \xrightarrow{r_A} & A_{X_0} \\ \downarrow{\scriptstyle \phi_{X_0}} & & \downarrow{\scriptstyle \phi_{X_1} \times \cdots \times \phi_{X_k}} \\ B_{X_1} \times \cdots \times B_{X_k} & \xrightarrow{r_B} & B_{X_0} \end{array}$$

Remark 1. (The following is an observation due to Lawvere in a paper written in his time teaching at Reed College [**lawvere_functorial_1963**].) The familiarity of this diagram is no mistake: indeed, by analogy to Theorem 2, we may understand algebras as product-preserving functors. A mapping between algebras then is a natural transformation, hence the naturality diagram in Definition 11. We will later make this analogy more concrete in Theorem 3.

Remark 2. Predictably, the $\mathfrak{C}$-valued algebras and homomorphisms of an algebraic theory $\mathcal{L}$ form a category, called $\mathcal{M}od_{\mathfrak{C}}(\mathcal{L})$.

Proof. Per Remark 1, we consider algebras and their homomorphisms as functors and natural transformations respectively. The identity morphisms are the identity natural transformations whose component morphisms are the identities of $\mathfrak{C}$. Composition of natural transformations is given by composition of their component morphisms, hence we may out-source the associativity condition to that guaranteed by the categorical structure of $\mathfrak{C}$. □

Most questions in type theory are concerned with the *terms* of the theory at hand. Normalization theorems talk about the accessibility (under some reduction relation) of a certain class of terms from any arbitrary term. Canonicity, a stronger property implying normalization, talks about the accessibility of another more strict class of terms from arbitrary start terms. These are but two examples of a broad spectrum of properties one might desire of the terms of a theory. Considering the primacy of term properties in type theory, it is rather strange that the notion of semantics we have built so far makes no commentary on terms besides the action on the clones given in Theorem 1. Our models so far have only given meaning to the individual *sorts* (types) and individual *operation symbols* (constructors) of the theory considered. In fact, this is enough: our models extend canonically to contexts and substitutions and thus give meaning to terms.

Definition 12 (Extending a model to terms)**.** Let A be an $\mathcal{L}$-algebra in a category $\mathfrak{C}$. This algebra extends canonically to an interpretation $[\![-]\!]$ of contexts by the following definition recursive in the structure of contexts:

$$[\![\emptyset]\!] = \mathbb{1}_C \tag{1.1}$$

$$[\![\Gamma, x : X]\!] = [\![\Gamma]\!] \times A_X \tag{1.2}$$

The (overly) careful reader will complain that $\mathfrak{C}$ doesn't necessarily feature a terminal object, but it turns out that a terminal object is guaranteed[5] by the finite product closure we imposed on $\mathfrak{C}$ in our definition of algebras. We are good to go.

[5] as the nullary finite product

Recalling more from the definition of an algebra, we know that A gives meaning to each operation symbol $Y_1, \dots, Y_k \vdash r : Z$ as a morphism $r_A : A_{Y_1} \times \cdots \times A_{Y_k} \to A_Z$ and gives meaning to each constant $c : Z$ by a morphism $1_{\mathfrak{C}} \to A_Z$. We can extend this uniquely to arbitrary terms in the context $\Gamma \equiv [\![x_1 : X_1, \dots, x_n : X_n]\!]$ by the following recursive definition:

$$[\![x_i]\!] : [\![\Gamma]\!] \equiv A_{X_1} \times \cdots \times A_{X_n} \xrightarrow{\pi_i} A_{X_i} \tag{1.3}$$

$$[\![c]\!] : [\![\Gamma]\!] \xrightarrow{<_!} \mathbb{1}_C \xrightarrow{c_A} A_Z \tag{1.4}$$

$$[\![r\,(u_1, \dots, u_k)]\!] : [\![\Gamma]\!] \xrightarrow{\langle [\![u_1]\!], \dots, [\![u_k]\!] \rangle} A_{Y_1} \times \cdots \times A_{Y_k} \xrightarrow{r_A} A_Z \tag{1.5}$$

where the $[\![u_i]\!]$ are the interpretations of the sub-expressions of the expression in the final line, π_i is the ith projection guaranteed to us by the universal property of products, and $<_!$ is the unique map into the terminal object. The angle bracket notion is used to express the product functor's action on morphisms in $\mathfrak{C}$. For clarity, we write out explicitly the composites for the reader:

$$\begin{aligned} [\![x_i]\!] &\equiv \pi_i \\ [\![c]\!] &\equiv c_A \circ <_! \\ [\![r\,(u_1, \dots, u_k)]\!] &\equiv r_A \circ \langle [\![u_1]\!], \dots, [\![u_k]\!] \rangle \end{aligned}$$

Theorem 3 (The classifying category of an algebraic theory)**.** *Let $\mathcal{L}$ be an algebraic theory. Then*

1. $\mathrm{Cn}^{\times}_{\mathcal{L}}$ *has finite products and an $\mathcal{L}$-algebra.*
2. *Let $\mathfrak{C}$ be another category with a choice of finite products and an $\mathcal{L}$-algebra. Then the functor $[\![-]\!] : \mathrm{Cn}^{\times}_{\mathcal{L}} \to \mathfrak{C}$ preserves finite products and the $\mathcal{L}$-algebra, and is the unique such functor.*
3. *Any functor $\mathrm{Cn}^{\times}_{\mathcal{L}} \to C$ which preserves finite products also preserves the mathcal$L$-algebra.*

Proof.

1. We first show that the syntactic category has finite products. Recall that the objects of the syntactic category are variable contexts $[x : X, y : Y, z : Z, \dots]$. For any other context $[t : T, u : U, v : V, \dots]$ we have the product

   $$[x : X, y : Y, z : Z, \dots] \times [t : T, u : U, v : V] = [x : X, y : Y, z : Z, \dots, t : T, u : U, v : V, \dots]$$

   That is, products are given by concatenation of contexts. Now the model is given as follows:

   (a) The sorts X of $\mathcal{L}$ are interpreted as single variable contexts $[x : X] \in \mathrm{ob}\, C$ where the variable x is arbitrary.

(b) The operation symbols $X_1, X_2, \cdots \vdash r : Y$ of $\mathcal{L}$ are interpreted as substitutions $[r\,(x_1, x_2, \ldots)\,/y] : [x_1 : X_1, x_2 : X_2, \ldots] \to [y : Y]$

2. The functor promised is precisely the one defined by Definition 12. TODO: we still need to verify the uniqueness of this functor, which Taylor himself never demonstrates.

3. This result demands a full proof, which isn't given in Taylor.

Proof. Suppose we have a functor $F : \mathrm{Cn}^{\times}_{\mathcal{L}} \to \mathfrak{C}$ which preserves products. We will show that it preserves the $\mathcal{L}$-model, in particular, that it takes the interpretation in $\mathrm{Cn}^{\times}_{\mathcal{L}}$ of any context Γ to the interpretation of Γ in $\mathfrak{C}$. TODO.

Note: Taylor claims the proof of this is given in [**lawvere_functorial_1963**], but I am (so far, I haven't tried too long) unable to identify the result in that paper, probably because Lawvere's formulation of algebraic theories is very different from Taylor's. I plan to tie up this loose end sometime soon. □

□

Chapter 2

Sketching the simply typed lambda calculus

Chapter 3

Normalization by evaluation: the classical perspective

In this chapter, we outline the technique of *normalization by evaluation* which exploits an existing programming language evaluator to produce a normalization function for the *open terms* of some other language. Our intent is merely to communicate the ideas involved with a view to drawing analogies to normalization by evaluation in our later discussion of gluing. In light of this, we describe the technique in full detail but omit all proofs of correctness (in the literature called "soundness") The chapter will follow our OCaml implementation of normalization by evaluation for Gödel's System T. Our implementation was inspired by the habilitation of Andreas Abel and the Wikipedia page on normalization by evaluation written by a number of authors.

Chapter 4

Normalization by gluing

> "i really thought u meant arts and crafts glue"
>
> new gluing understander

This chapter is the culmination of all that we've developed so far in the thesis. We will consider the normalization problem for the simply typed lambda calculus. Normalization comes in both weak and strong flavors, but both are properties of a formal system together with a *reduction* relation, say, $- \rightarrow =$. Now by a *normal form* we mean term t for which there is no other t' such that $t \rightarrow t'$. Writing $- \rightarrow^* =$ for the least transitive, reflexive closure of this relation, to say that the reduction system enjoys *weak normalization* means that for any well-typed term e, there exists a normal form n such that $e \rightarrow^* n$. A reduction system enjoying *Strong normalization* is one for which *every* reduction sequence ends in a normal term; equivalently, *strong normalization* requires that there are no infinite reduction sequences over distinct terms. *Metatheorems* of this kind tend to resist standard techniques like straightforward induction over the types of the lambda calculus. Instead, metatheoreticians resort to the *occult* technique of logical relations. (cite TAPL). The method of logical relations is, broadly speaking, somewhat opaque even to experienced practitioners. The construction is simple to write down and use in easy settings like that of the simply-typed lambda calculus, but becomes devilishly complicated when working with more complex lambda calculi.

The method of *gluing* taken from the category theorists book is deeply related to the method of logical relations, but regularizes some of the thinking involved and thereby makes most of the choices involved in the usual proofs by logical relations completely automatic. The method of gluing is well-known in mathematics, but some of its instances are more famous than others. The *scone* or *Sierpinski cone* is a famous instance of the gluing construction used in and outside of type theory. The *scone* is not suited to our particular problem however, as the scone is generally concerned with properties of *closed terms* with no free variables, whereas our construction will consider terms in any context.

4.1 Variable-arity Kripke relations

4.2 The comma construction and friends

4.2.1 An easy example: the category of renamings

The category of renamings is, in a very loose sense, like a less proof relevant version of the category of contexts and substitutions. In particular, if there exists a substitution relating two contexts in syntactic category, then there exists a renaming relating the same contexts. In this sense, renamings allow us to track changes in context due to substitution without actually using substitutions which are problematic due to their being overly quotiented by definitional equality thanks to the substitution lemma. The category of renamings allows us to define the presheaves we need while not losing track of how contexts evolve by substitution. I think that these two points are precisely the desiderata satisfied by the category of renamings.

Need to present its

1. Definition as a comma: a context is a finite set of variables paired with an assignment of types to each variable; renamings are those functions of the underlying variable sets which preserve the extra globbed on typing information

2. Straightforward definition

Kripke relations indexed by renaming contexts

Should just provide the instance we'll be working with here to lower the reader's cognitive burden of too much generality.

4.2.2 Presheaves of syntax, defined over the category of renamings

A lambda-algebra in Ren-hat

4.2.3 Why the syntactic category just won't do

talk about why neutrals and normals can't support a substitution action

Remark 3 (Any category of presheaves are cartesian closed). This is a well-known fact. Maybe I can bloviate on this a smidge. I think it goes well in this subsection.

4.2.4 A harder example: the gluing category; or, the category of proof-relevant Ren-Kripke relations over types

This subsection introduces Fiore's monster.

The relative hom functor

Gluing semantics to syntax along the relative hom functor

4.2.5 The subcategory of (ordinary) variable-arity Kripke relations

We can recover ordinary, proof-*irrelevant* variable-arity Kripke relations as a subcategory of the gluing category. In particular, ordinary variable-arity Kripke relations arise as those objects $\left(D : \widehat{\mathrm{Ren}_\Sigma}, q : D \Longrightarrow \mathfrak{Tm}\left(\Delta\right), \Delta : \mathrm{Cl}_\Sigma\right)$ of the gluing category whose quotient map q is a component-wise monomorphism; i.e., for each $\Gamma \in \mathrm{Ren}_\Sigma$, we have that $q_\Gamma : D\left(\Gamma\right) \rightarrowtail \mathfrak{Tm}\left(\Delta\right)\left(\Gamma\right)$ is a monomorphism. Here's why. Recalling the definition of Ren_Σ-Kripke relations over $\tau \in \mathrm{Cl}_\Sigma$, we need to find in these data a Ren_Σ-indexed family $\left\{R_\Gamma\right\}_{\Gamma \in \mathrm{Ren}_\Sigma}$ of sets of Cl_Σ-morphisms into τ subject to the following relative monotonicity condition: for any renaming $\rho : \Gamma' \to \Gamma$, if t TODO. Since we're working in $\mathfrak{Set}$, those monomorphisms are the usual injective functions. To see why this recovers what we had before, we need to look at the naturality diagram of q:

$$\begin{array}{ccc} R\left(\Gamma\right) & \xrightarrowtail{q_\Gamma} & \mathfrak{Tm}\left(\Delta\right)\left(\Gamma\right) \\ \downarrow{\scriptstyle R(\rho)} & & \downarrow{\scriptstyle \mathfrak{Tm}(\Delta)(\rho)} \\ R\left(\Gamma'\right) & \xrightarrowtail{q_{\Gamma'}} & \mathfrak{Tm}\left(\Delta\right)\left(\Gamma'\right) \end{array} \qquad \begin{array}{c} t \\ \downarrow \\ t \circ \rho \end{array}$$

Because they are monos, we may allow ourselves to identify the component morphisms with their images. In particular, we write $\left|q_\Gamma\right| \subseteq \mathfrak{Tm}\left(\Delta\right)\left(\Gamma\right)$ for the image for each Γ. Now what this diagram says is that for any $q_\Gamma\left(r\right) = t \in \left|q_\Gamma\right|$, we have that $t \circ \rho = q_{\Gamma'}\left(R\left(\rho\right)\left(r\right)\right) \in \left|q_{\Gamma'}\right|$. This is precisely the relative monotonicity condition for $\mathfrak{C}$-Kripke relations: containment in each predicate is functorial with respect to renaming *up to renaming*.

4.3 Obtaining a normalization function

4.3.1 A lambda-algebra for the gluing category

4.3.2 Cartesian closed structure for the gluing category

4.3.3 A promise kept: normalization

Chapter 5

Conclusion

That's it for now.

Appendix A

The First Appendix

Appendix B

The Second Appendix, for Fun