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.

Notation

Following the great masters[1], the face $\mathfrak{Mathfrak}$ is used liberally and wherever possible without any particular convention.

[1] Kale Ormsby

Table of Contents

Introduction **1**

Chapter 1: Functorial semantics **3**
1.1 An algebraic prelude 4
1.2 Elementary sketches and their models 4
1.3 The category of contexts and substitutions 7
1.4 Models are essentially functors 9
1.5 Algebraic theories and their algebras 9

Chapter 2: Syntax and (functorial) semantics of the lambda calculus **15**
2.1 The simply typed lambda calculus 15
2.2 Raw CCS and beta-eta is exactly CCS 16
2.3 An algebraic treatment of the lambda calculus 19

Chapter 3: Normalization by gluing **23**
3.1 The comma construction and friends 24
3.1.1 An easy comma: the category of renamings 24
3.1.2 Variable-arity Kripke relations 26
3.1.3 A harder example: the gluing category; or, the category of proof-relevant Ren-Kripke relations over types 27
3.2 Presheaves of stratified neutral and normal syntax 29
3.2.1 Variables and binding in Renhat 30
3.2.2 Stratifying neutral and normal terms 30
3.2.3 Presheaves of syntax over the category of renamings 33
3.2.4 A stratified lambda-algebra of neutrals and normals in Renhat 34

Chapter 4: Obtaining a normalization function **37**
4.1 Desiderata for a normalization function 37
4.2 Cartesian closed structure for the gluing category 38
4.3 Gluing syntax to semantics 42
4.4 Taking stock: charting a path to normalization 46
4.5 Glued reification and reflection 47
4.6 A promise kept: a function from open terms to normal terms 51
4.7 Reaping what we've sown: easy proofs of the correctness properties . 51
4.8 Looking forward: "blah" by gluing 52

Conclusion **53**

List of Tables

List of Figures

Abstract

Normalization by gluing is based.

Introduction

> You don't need a weatherman to
> know which way the wind blows.
>
> Bob Dylan

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: sketches and their models; algebraic theories and their algebras

In this chapter we will develop tools for reasoning about (syntactic) *theories*, which are "notions of" abstract structure. As an example, this chapter will present a theory encoding the structure of a ring (as in algebra.) A more complicated example will come in Chapter 2 which will apply the ideas of this chapter to 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. We will more laconically refer to a notion of syntactic theory as a *doctrine*. A *doctrine* is something like a 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 specify theories involving *unary* operations (those defined over a single parameter.) 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 parameters, thus covering a broad variety of gadgets one might encounter in mathematics and computing. Algebraic theories are known more famously as *Lawvere theories* after categorical logic superstar (and former Reed College professor!) William Lawvere who originally presented them while building his functorial treatment of universal algebra. Emphasizing the doctrinal upgrade, some authors refer to algebraic theories as *finite product sketches* [**wells_sketches_2009**] though we will tend to stick with Lawvere's original terminology. 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; indeed, much of what follows draws heavily from Paul Taylor's *Practical Foundations of Mathematics* [**taylor_practical_1999**]. My contribution is to flesh out some of the examples found there, add some of my own, and make other parts of the presentation more palatable and quickly digestible to the reader already acquainted with basic category theory and type theory.

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 1.1.1. Recall that a **covariant action** of a group or monoid $(M, id, \cdot)$ on a set A is a function $(-)_* (=) : M \times A \to A$ 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 size 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 1.1.2. A **faithful** action $(-)_*$ is one for which we have

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

for all f, g in M. Morally, this requirement says that elements are *semantically* equal (or, act the same) only when they are *syntactically* equal (or, *look* identical.)

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.

1.2 Elementary sketches and their models

As promised, we begin with the definition.

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

Definition 1.2.1. An **elementary sketch** is comprised of the following data:

1. a collection $X, Y, Z, \ldots$ 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 1.2.2 (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 concept which makes more concrete what we mean when saying "generates." The idea should be familiar to the reader acquainted with type theory, despite the drastically simplified setting.

Definition 1.2.3. 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 1.2.4 (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. It is well-known that substitution operation is associative; see, for example, Chapter 1 of [**taylor_practical_1999**] for a fantastic treatment of the theory of subtitution. As a consequence, any elementary sketch, including this one, satisfies the associativity axiom of monoids for free. The identity term is given by zero composable unary operation-symbols (i.e., the empty string) applied to a variable. Because composition is given by substitution for the variable, a lone variable does indeed work function as the identity. Freedom comes from the fact that the sketch has no equations.

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 1.2.5. 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

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

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 also prove immediately useful in studying Example 1.2.2 by forming the sets of a "for-free" model for any elementary sketch.

Definition 1.2.6. 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 1.2.7. Every elementary sketch has a faithful covariant action on its clones $\mathcal{H}_X = \cup_\Gamma \mathrm{Cn}_{\mathcal{L}}(\Gamma, X)$ by sequencing with the operation symbol. Substitution for the (single) variable in a term gives a faithful contravariant action on $\mathcal{H}^Y = \cup_\Theta \mathrm{Cn}_{\mathcal{L}}(Y, \Theta)$.

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:

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

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 1.2.2, 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 1.2.8. A couple of type theorists walk into a Michelin starred restaurant. The menu reads in blackboard bold letters "$\mathbb{NO}\ \mathbb{SUBSTITUTIONS}$". They promptly leave.

The reader might have noted that there is a strong resemblance between the notion of a sketch and that of a category. Indeed, the following result uses the clone action defined above to show that we can associate to every sketch a category and that to every category we can associate a sketch. These assignments moreover induce an isomorphism of categories.

Theorem 1.2.9 (The canonical elementary language)**.** Every elementary sketch $\mathcal{L}$ presents a category $\mathrm{Cn}_{\mathcal{L}}$ via the for-free action on its clones, and conversely any small category C is presented by some sketch $\mathcal{L}$ in the sense that $C \cong \mathrm{Cn}_{\mathcal{L}}$. We write $\lceil - \rceil$ for this isomorphism and call the sketch $L(C) = \mathcal{L}$ the *canonical elementary language* of C.

The language is defined as follows:

- The sorts $\lceil X \rceil$ of $L(C)$ are the objects X of C
- The operation symbols $\lceil f \rceil$ are the morphisms f of C and
- The laws are $\lceil \mathrm{id} \rceil (x) = x$ and $\lceil g \rceil \left(\lceil f \rceil (x) \right) = \lceil g \circ f \rceil (x)$

and the isomorphism $C \cong \mathrm{Cn}_{\mathcal{L}}$ is clear.

1.3 The category of contexts and substitutions

We now introduce a category which is special in both the structure it enjoys as well as the central role it will play in the rest of this thesis. This category goes by many names: *syntactic category*, the 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 1.3.1 (The category of contexts and substitutions)**.** Given a sketch $\mathcal{L}$, the **category of contexts and substitutions**, written $\mathrm{Cn}_{\mathcal{L}}$ 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* $\left[a/x\right] : \Gamma \to [\Gamma, x : X]$ 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* $\hat{x} : [\Gamma, x : X] \to \Gamma$ for each variable $x : X$.
- The laws are given by an extended version of the substitution lemma from type theory [**pierce_types_2002**]. When talking about these laws, we will often call them "the extended substitution lemma." The following laws are imposed 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:

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

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 change of base functor is contravariant, this requires considering the compositions in reverse order as:

$$\hat{y}^*; \left[y/x\right]^* ; \hat{x}^*; \left[x/y\right]^* = \mathrm{id}^*$$

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.

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

This category allows 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.

1.4 Models are essentially functors

Theorem 1.4.1 (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 1.2.9) 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 follows from the laws of the canonical elementary language and faithfulness of the model.

($\Leftarrow$) 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 1.2.9 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 $\lceil g \rceil \left(\lceil f \rceil (x) \right) = \lceil g \circ f \rceil (x)$. 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 $\left(F_A \lceil g \rceil\right) \circ \left(F_A \left(\lceil f \rceil \right)\right) = F_A(g \circ f)$. Both are ensured by functoriality.

□

1.5 Algebraic theories and their algebras

While conjuring up example elementary sketches, the reader will find that the unary operation requirement regularly gets in the way of representing familiar gadgets like

rings or type theories. To address this deficiency, we introduce the doctrine of *algebraic theories* which generalizes the doctrine of elementary sketches and allows for multi-input operation symbols. As we work through this upgrade in doctrine, many of the notions (*terms, clones, syntactic category, etc.*) we developed in the simplified world of elementary sketches will come along for the ride without much changed.

Definition 1.5.1 (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**, $x_1 : X_1, \ldots, x_k : X_k \vdash r\,(x_1, \ldots, x_k) : Y$ 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.

Example 1.5.2 (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 $\{\,\square\,\} \cup \{\,\square_i\,\}_{i\in\mathbb{N}}$ and $\{x, y, z\} \cup \{\,s_i\,\}_{i\in\mathbb{N}}$ 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:

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

Note that the countably infinite sets of variables $\{\square\}_{i\in\mathbb{N}}$ and $\{s\}_{i\in\mathbb{N}}$ are added so that we may have deeply nested expressions, even if we had no need for them when stating the laws above. In fact, such countable variable sets are *required* when defining an algebraic theory for precisely this reason. That this requirement was not imposed for elementary sketches is because there can only be one variable in a term, even heavily nested ones, by construction.

Having given the obligatory concrete example of a theory, we now allow ourselves another abstract definition: that of an $\mathcal{L}$-*algebra* for an algebraic theory:

Definition 1.5.3 ($\mathcal{L}$-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 1.5.2.

Example 1.5.4. 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 function $* \mapsto 0 \in \mathbb{Z}$
   (d) 1_A to the constant function $* \mapsto 1 \in \mathbb{Z}$
   (e) $-_A$ to the function $x \mapsto -x$ taking an integer to its additive inverse
3. Verifying the rest of the laws is routine after understanding how to verify any of them, 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.

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

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 1.5.8 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 $\hat{\square}^* x$ by[3] 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 amounts to verifying that the integers form a ring under the standard multiplication and addition operations we learn in elementary school. In general, the reader will find that all *rings* (in the conventional sense of abstract algebra) are $\mathfrak{Set}$-valued *algebras* of the algebraic theory of *ring*. The reader may also verify that the usual ring-homomorphisms in $\mathfrak{Set}$ are an instance of the following generalized notion of homomorphism:

Definition 1.5.5 ($\mathcal{L}$-algebra homomorphism)**.** A *homomorphism* $\phi : A \to B$ 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:

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

Remark 1.5.6. 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 1.4.1, we may understand algebras as product-preserving functors. A mapping between algebras then is a natural transformation, hence the naturality diagram in Definition 1.5.5. We will later make this analogy more concrete in Theorem 1.5.9.

Remark 1.5.7. Perhaps predictably, the $\mathfrak{C}$-valued algebras and homomorphisms of an algebraic theory $\mathcal{L}$ form a category, called $\mathrm{Mod}_{\mathfrak{C}}\left(\mathcal{L}\right)$.

[3] recall from Definition 1.3.1 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

Proof. Per Remark 1.5.6, 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 $\mathfrak{C}$. □

Many things you'd like to prove about a 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 open term. *Canonicity* or *closed normalization* theorems talk about the accessibility of a different class of terms from arbitrary *closed* 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 1.2.7. 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 *along the product structure* to contexts and substitutions and thus give meaning to terms.

Definition 1.5.8 (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:

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

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

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[4] by the finite product closure we imposed on $\mathfrak{C}$ in our definition of algebras. We are good to go.

Recalling more from the definition of an algebra, we know that A gives meaning to each operation symbol $Y_1, \ldots, Y_k \vdash r : Z$ 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 $\mathbb{1}_{\mathfrak{C}} \to A_Z$. We can extend this to arbitrary terms in the context $\Gamma \equiv [\![x_1 : X_1, \ldots, x_n : X_n]\!]$ by the following recursive definition:

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

$$[\![c]\!] : [\![\Gamma]\!] \xrightarrow{<_!} \mathbb{1}_{\mathfrak{C}} \xrightarrow{c_A} A_Z$$

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

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

[4] as the nullary finite product

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 1.5.9 (The classifying category of an algebraic theory)**.** Let $\mathcal{L}$ be an algebraic theory. Then $\mathrm{Cn}_{\mathcal{L}}$ has finite products and

1. there exists in $\mathrm{Cn}_{\mathcal{L}}$ an $\mathcal{L}$-algebra which satisfies the following universal property:
2. Let $\mathfrak{C}$ be another category with finite products and its own an $\mathcal{L}$-algebra. Then the functor $[\![-]\!] : \mathrm{Cn}_{\mathcal{L}} \to \mathfrak{C}$ preserves finite products and the $\mathcal{L}$-algebra, and is the unique such functor.

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

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

That is, products are given by concatenation of contexts. The projections are given by weakening by all the variables in either context: that is, the second projection is $\Gamma \times \Delta \xrightarrow{\hat{\gamma_1} \circ \cdots \circ \hat{\gamma_k}} \Delta$ where the γ_i are the variables of Γ. We define the first projection similarly. The universal property for products is upheld by the equational laws concerned with weakenings in the category of contexts.

Now come the more interesting promised results:

1. (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 $\left[r\,(x_1, x_2, \dots)\,/y\right] : [x_1 : X_1, x_2 : X_2, \dots] \to [y : Y]$
2. The functor promised is precisely the one defined by Definition 1.5.8. Preservation of the model and of finite products both follow from the definition of the interpretation by induction on contexts. Uniqueness of the functor is in turn forced by preservation of the model and finite products.

□

Chapter 2

Functorial semantics of the simply typed lambda calculus in cartesian closed categories

"Eeny, meeny, miny, moe"

Alonzo Church (Allegedly, on his choice of λ as the name for his calculus.)

This chapter exploits the heavy machinery developed in the previous chapter to give meaning, in specially structured categories, to the types and terms of the simply typed lambda calculus. Here is how we shall do so: First, we will present various a suite of simple type theories as a series of *algebraic theories* differing only in their equational laws. One of these type theories will be the typical simply typed lambda calculus with the full gamut of equational laws, including β-conversion and η-conversion and α-renaming. Second, we present an interesting perspective on a well known type of categorical structure, namely *cartesian closedness*, by explicitly defining it in terms of the definitional β and η laws from type theory. We will find that this characterization exactly coincides with the one found in typical books on category theory. Finally, we will show that any interpretation of base types of the lambda calculus[1] gives rise to an interpretation of lambda terms in any *cartesian closed* category with an interpretation of the base types. Unless otherwise noted, what follows is an adaptation of [**taylor_practical_1999**] (Chapter 4.7). We begin by presenting the term language and type system for the calculus under consideration.

2.1 The simply typed lambda calculus

We'll work with the simply typed lambda calculus with some collection T of *base types* (say natural numbers, integers, booleans, or some other type of object the reader is

[1] as above

interested in.) We will not dwell on the details of standard aspects of this development and instead refer the reader to the standard references ([**pierce_types_2002**], [**harper_practical_2016**]) for the full story.

Definition 2.1.1 (Types)**.** Given a set of base types T, the types $\widetilde{T}$ of the simply typed lambda calculus are generated by the following grammar:

$$\tau ::= \mathbb{1} \mid \tau_1 \to \tau_2 \mid \tau_1 * \tau_2 \mid \theta$$

where θ is drawn from T. Note that the unit type $\mathbb{1}$ is not a base type but rather can be considered a nullary product of types.

Definition 2.1.2 (Terms)**.** The terms of the simply typed lambda calculus (with booleans) are generated by the following grammar:

$$t ::= x \mid () \mid \pi_1 t \mid \pi_2 t \mid (t_1, t_2) \mid t_1 t_2 \mid \lambda x : \tau . t$$

where the variables x are drawn from a countably infinite set V.

The typing rules are given as follows:

Definition 2.1.3 (Typing rules)**.**

$$\frac{\Gamma \vdash t : \tau_1 * \tau_2}{\Gamma \vdash \pi_i t : \tau_i} \qquad \frac{\Gamma \vdash t_i : \tau_i}{(\tau_1, \tau_2) : \tau_1 * \tau_2}$$

$$\frac{\Gamma \vdash t_1 : \tau' \to \tau \qquad \Gamma \vdash t_2 : \tau'}{\Gamma \vdash t_1 t_2 : \tau} \qquad \frac{\Gamma, x : \tau' \vdash t : \tau}{\Gamma \vdash \lambda x : \tau' . t : \tau' \to \tau}$$

$$\frac{}{\Gamma \vdash x : \tau} \ (\mathrm{x}{:}\tau) \ \in \Gamma$$

$$\frac{}{\Gamma \vdash () : \mathbb{1}}$$

2.2 Raw cartesian closed structure, beta-eta rules, and cartesian closed structure

The following notion mediates between *having lambda abstraction* and the more structured situation enjoyed by usual presentations of the lambda calculus. The mediating notion, namely *raw cartesian closed structure*, merely requires that one be able to write down lambda abstractions, and that they behave nicely with respect to substitutions that don't interfere with the bound variable. Notably lacking in *raw cartesian closed structure* from the more structure situation referred to above are the *β- and η-laws* which explain how to *compute* with lambda abstraction.

The definition comes from Taylor [**taylor_practical_1999**].

Definition 2.2.1 (Raw cartesian closed structure)**.** Let $\mathfrak{C}$ be a category with a specified terminal object $\mathbb{1}$ and specific products together product projections. We say that $\mathfrak{C}$ is *raw cartesian closed* or *has raw cartesian closed structure* if we have the following for each pair of objects X, Y of $\mathfrak{C}$:

1. An object Y^X
2. An morphism, *application*, $\mathrm{ev}_{X,Y} : Y^X \times X \to Y$ and
3. For each object Γ, a function of hom-sets $\lambda_{\Gamma,X,Y} : \mathfrak{C}(\Gamma \times X, Y) \to \mathfrak{C}\left(\Gamma, Y^X\right)$ obeying the naturality law:

   $$\lambda_{\Gamma,X,Y}\left(\mathfrak{p} \circ (\mathfrak{u} \times \mathrm{id}_X)\right) = \lambda_{\Gamma,X,Y}(\mathfrak{p}) \circ \mathfrak{u}$$

   for each $\mathfrak{u} : \Gamma \to \Delta$ and $\mathfrak{p} : \Delta \times X \to Y$.

   Looking closely at the naturality law, it essentially says that substitutions "not touching" the bound variable commute with lambda abstraction. Informally, this is a good equation to have in mind: $\mathfrak{u}^*(\lambda x\,.p) = \lambda x.\ (\mathfrak{u}^* p)$.

We will say that an algebraic theory $\mathcal{L}$ *has raw cartesian closed structure* if its classifying category $\mathrm{Cn}^{\times}_{\mathcal{L}}$ has raw cartesian closed structure.

The following theorem extends Definition 1.5.8 and shows how to construct an interpretation of terms from an algebra in a raw cartesian closed category. The essence is that we extend the assignments of base types and operation symbols along products (as we did before when dealing with ordinary algebraic theories) and along the new exponential objects as well.

Theorem 2.2.2. Let $\mathcal{L}$ be an algebraic theory and C be a category with raw cartesian closed structure and an algebra for $\mathcal{L}$. The algebra for $\mathcal{L}$ extends uniquely along the raw cartesian closed structure to provide an interpretation of terms, which defines a functor $[\![-]\!] : \mathrm{Cn}_{\mathcal{L}} \to C$

Proof. Building on Definition 1.5.8, we define the interpretation by structural recursion:

- The interpretation of base types are given by the algebra for $\mathcal{L}$
- The interpretation of an exponential Δ^{Γ} is $[\![\Delta]\!]^{[\![\Gamma]\!]}$ where the latter is the exponential entailed by the raw cartesian closed structure.
- Contexts are taken to products as in Definition 1.5.8.
- The variables, operation symbols, and the laws are treated as in Definition 1.5.8.
- The last clause of the raw cartesian closed structure gives the notion of lambda abstraction in C.

□

Definition 2.2.3 (Beta-eta rules)**.** A raw cartesian closed algebraic theory $\mathcal{L}$ satisfies the $\beta - \eta$ *rules* if for all $\mathfrak{p} \in \mathrm{Cn}_{\mathcal{L}}(\Gamma \times X, Y)$ we have the equations

$$\mathrm{ev}_{X,Y} \circ \left(\lambda_{\Gamma,X,Y}(\mathfrak{p}) \times \mathrm{id}_X\right) = \mathfrak{p} \qquad (\beta)$$
$$\lambda_{Y^X,X,Y}\left(\mathrm{ev}_{X,Y}\right) = \mathrm{id}_{Y^X} \qquad (\eta)$$

These equations deserve some commentary. The beta rule says that application of an abstracted body $\mathfrak{p}$ to the identity gives you back the body you started with: the beta rule forces that application of lambda expressions does nothing more than substitute for the abstracted variable. The eta rule says that taking a function f, applying it to a variable x, then finally abstracting over that variable x is the same operation as doing nothing to f. We will see the more familiar symbolic forms of these rules in the definition of the theory of the usual calculus below.

We now define a notion more well-known outside of computing. Cartesian closed structure endows a category with the ability to take products and function spaces over its objects in a suitable fashion; as can be seen below, the notion of Cartesian product and function space in the category of sets are examples of suitable such constructions. It will turn out that this familiar situation is equivalent to the combination of raw cartesian closed structure and beta-eta rules.

Definition 2.2.4 (Cartesian closed structure)**.** Let $\mathfrak{C}$ be a category. An *exponential* of objects X and Y is an object E of $\mathfrak{C}$ together with a morphism $\epsilon : E \times X \to Y$ such that for each object Γ of $\mathfrak{C}$ and morphism $p : \Gamma \times X \to Y$ there is a unique morphism $\tilde{p} : \Gamma \to E$ called the *exponential transpose* such that $p = \epsilon \circ (\tilde{p} \times \mathrm{id})$. This is a universal property: exponentials are unique up to unique isomorphism. A category is *cartesian closed* if it has all products and exponentials.

The category $\mathfrak{Set}$ is the prototypical cartesian closed category whose products are the usual cartesian products of sets and whose exponentials are function spaces: for sets X and Y, $Y^X = \left\{\text{functions } f \mid \mathrm{dom}(f) = X \wedge \mathrm{cod}(f) = Y\right\}$. Another good source of example cartesian closed categories is topos theory. All elementary toposes, including categories of presheaves, are cartesian closed [**leinster_informal_2011**].

It turns out that all of these examples of cartesian closed categories can also be characterized as raw cartesian closed categories that satisfy the beta-eta rules.

Theorem 2.2.5 (Finite-product category + raw CCS + beta-eta $\iff$ cartesian closed)**.** Let $\mathfrak{C}$ be a category. Then $\mathfrak{C}$ is cartesian closed if and only if $\mathfrak{C}$ has finite products, a specified terminal object, and both is raw cartesian closed and satisfies the beta-eta laws.

We will not repeat Taylor's proof [**taylor_practical_1999**], but we will say that the proof largely comes down to these two facts: that exponential transposes are unique and that the characterizing proprety of the transpose is exactly the β rule.

2.3 An algebraic treatment of the lambda calculus

Finally, we will present the lambda calculus as a series of algebraic theories, each of which are identical except for the equations they admit. These theories correspond to instances of the lambda calculus whose respective definitional equalities identify more or fewer lambda terms. We will see that algebras of these theories are essentially functors valued in specially structured categories.

Definition 2.3.1 (Algebraic theory of the lambda calculus)**.** The *α-lambda calculus* is presented as the following algebraic theory called $\mathcal{L}_{\lambda\alpha}$:

1. Sorts: We define the sorts to be the set $\widetilde{T}$ given in Definition 2.1.1.

2. Variables: The variables are defined as the collection $\left\{v : \tau \mid (v, \tau) \in V \times \widetilde{T}\right\}$ where V is the countably infinite set V of untyped variables as in Definition 2.1.2.

3. We define the following operation symbols for each $\tau_1, \tau_2, \tau, \tau' \in \Sigma_\lambda$:

$$t : \tau_1 * \tau_2 \vdash \pi_1(t) : \tau_1$$
$$t : \tau_1 * \tau_2 \vdash \pi_2(t) : \tau_2$$
$$[t_1 : \tau_1, t_2 : \tau_2] \vdash (t_1, t_2) : \tau_1 * \tau_2$$

$$\left[t : \tau' \to \tau, t' : \tau'\right] \vdash t\, t' : \tau$$

$$\vdash () : \mathbb{1}$$

   The reader will note that we have not yet added an operation symbol for lambda abstraction. This requires careful consideration, because lambda abstraction is not really an operation symbol but rather a family of operation symbols defined mutually with the others. For each term $\Gamma, x : \tau' \vdash t : \tau$, we add a new operation symbol

$$\Gamma \vdash (\lambda x.\, t)\left(\overrightarrow{\gamma}\right) : \tau$$

   where $\overrightarrow{\gamma}$ is the list of variables comprising Γ.

   Finally, the complete collection of operation symbols is defined as the least set closed under taking abstractions in this way and containing the operation symbols already mentioned above.

4. The only equation is the α-rule which says that terms can be identified up to renamings of their bound variables.

$$\lambda(x : X).\, t = \lambda\left(x' : X\right).\, \left[x'/x\right]^* t \qquad (\alpha)$$

defined for each operation symbol $f : Y^X$ and terms $\Gamma, x : X \vdash t : \tau$ and $\Gamma \vdash t' : X$.

Note that we do not include the $\beta\eta$-rules in this version.

With this theory in hand, we first show that it is raw cartesian closed (i.e. that it supports a reasonably coherent notion of lambda abstraction.) After that, we will add to the theory the β and η laws. Finally, we will use the preceding theorems to argue *from β and η* to show that the resulting theory of the typical lambda calculus is cartesian closed.

Theorem 2.3.2 (α-lambda calculus is raw cartesian closed)**.** The theory $\mathcal{L}_{\lambda\alpha}$ is raw cartesian closed

Proof. In the following we write $[X \Rightarrow Y]$ for the exponential Y^X, for the sake of legibility. The exponential

$$\Big[\big[x_1 : X_1, x_2 : X_2 \ldots, x_j : X_j\big] \Rightarrow [y_1 : Y_1, y_2 : Y_2, \ldots, y_k : Y_k]\Big]$$

is the context $[f_1 : F_1, f_2 : F_2, \ldots, f_k : F_k]$ where the f_i are new variables and

$$F_i = X_1 \Rightarrow \Big[X_2 \Rightarrow \cdots \big[X_j \Rightarrow Y_i\big] \cdots \Big]$$

Finally, we define

$$\mathrm{ev}_{\vec{X},\vec{Y}} = \Big[\Big(\cdots \big(f_1\,(x_1)\,x_2\big) \cdots x_j\Big)\,/y_1\Big] \circ \cdots \Big[\Big(\cdots \big(f_k\,(x_1)\,x_2\big) \cdots x_j\Big)\,/y_k\Big]$$

$$\lambda_{\Gamma,\vec{X},\vec{Y}}\left[\overrightarrow{p}\,/\,\overrightarrow{y}\right] = \Big[\Big(\lambda x_1.\ \big(\lambda x_2. \ldots (\lambda x_k.\, p_1) \ldots\big)\Big)\,/f_1\Big] \circ \cdots \Big[\Big(\lambda x_1.\ \big(\lambda x_2.\ \ldots (\lambda x_k.\, p_k) \ldots\big)\Big)\,/f_k\Big]$$

Naturality with respect to weakening and single substitution follows from the fact that substitution for (or weakening by) a variable x commutes with abstraction of a distinct variable y: for any terms p and c, we have $\big[c/y\big]^*\,(\lambda x.\, p) \equiv \lambda x.\ \Big(\big[c/y\big]^*\,p\Big)$. The same situation holds for weakening. The full naturality law follows from this and the fact that single substitution and weakening are the generating morphisms of the classifying category. □

Definition 2.3.3 (Lambda calculus)**.** The lambda calculus, or the $\alpha\beta\eta$-lambda calculus is defined as the algebraic theory of the α-lambda calculus enriched with the following additional equational laws:

$$\big(\lambda\,(x : X)\,.\,t\big)\ t' = \big[t'/x\big]^*\,t \qquad (\beta)$$

$$f = \big(\lambda\,(x : X)\,.\,f\,x\big) \qquad (\eta)$$

We write $\mathcal{L}_{\lambda\alpha\beta\eta}$ or simply λ for the new theory, and Cn_λ for its classifying category.

Definition 2.3.4 (Lambda algebra)**.** By a *lambda algebra*, we mean an algebra for the theory $\mathcal{L}_{\lambda\alpha\beta\eta}$

The following results quickly follow the definition of the theory of the lambda calculus and the work we did in Chapter 1.

Theorem 2.3.5. The lambda calculus is a raw cartesian closed algebraic theory with beta-eta.

Theorem 2.3.6. The syntactic category Cn_λ of the theory $\mathcal{L}_{\lambda\alpha\beta\eta}$ is cartesian closed and has a lambda algebra satisfying the following universal property: if C is a cartesian closed category with a lambda algebra then there is a unique functor $[\![-]\!] : \mathrm{Cn}_\lambda \to C$ that preserves the cartesian closed structure and the lambda algebra.

Proof. Cartesian closure follows immediately from the previous result and Theorem 2.2.5. The rest is given essentially as in Theorem 1.5.9, except that we must handle the new exponentials with more care. Theorem provides the requisite care and extends the functor defined in Definition 1.5.8 to provide one which preserves exponentials as required. □

This final result will prove important in Chapter 3.

Chapter 3

Normalization by gluing

> You, you were like glue
> holding each of us together
> I slept through July
> while you made lines in the heather
>
> Fleet Foxes, "Lorelai"

This chapter will consider the normalization problem for the simply typed lambda calculus using all of the magic technology we've developed so far. Normalization comes in both weak and strong flavors, but both are properties of a formal system together with a *reduction* relation, say, $- \to =$. By a *normal form* we mean a term t for which there is no other t' such that $t \to t'$. Writing $- \to^* =$ for the least transitive, reflexive closure of this relation, we say the reduction system enjoys *weak normalization* if for any well-typed term e, there exists a normal form n such that $e \to^* n$. A reduction system enjoying *Strong normalization* is one for which *every* reduction sequence ends in a normal term. *Metatheorems* of this kind tend to resist standard techniques like straight-forward induction over the types of the lambda calculus. Instead, metatheoreticians resort to the *occult* technique of logical relations. The method of logical relations is, broadly speaking, opaque even to experienced practitioners. The construction is easy to use in simple settings like that of the simply-typed lambda calculus (see [**pierce_types_2002**]) but becomes devilishly complicated when working with more complex lambda calculi.

The method of *Artin gluing*, torn from the topos theorist's cookbook, is a more transparent encoding of the mechanics of logical relations. Where the method of logical relations is ad-hoc and mysterious, gluing regularizes the thought involved and makes the choices involved in proofs by logical relations completely automatic. 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 of *(open) normalization* however, as the scone is generally concerned with properties of *closed terms* with no free variables, whereas our construction will apply to terms in any context.

3.1 The comma construction and friends

This section introduces a general construction that will, among other things, allow us to define the gluing construction from which this chapter gets its name. The loose idea of the comma is to glob some extra data to the objects of a category in a way that supports a well-defined notion of an object-with-extra-data-morphism which preserves the attached data. This intuition perhaps underlies the name for the gluing construction. Our specific use of the gluing construction will involve gluing presheaves of syntactic terms (as you would write them down on paper) together with presheaves of semantic terms (which are *substitutions*.) That is, the extra data we glob on is the semantic of terms, and the notion of extra-data-preserving morphism is a pair of a syntactic transformation and a substitution which preserve the semantic interpretation of the syntax.

Definition 3.1.1 (Comma category)**.** Let E, D, C be categories and $F : D \to C \leftarrow E : G$ be a pair of functors sharing their codomain C. The comma category $F \downarrow G$ has as its

1. Objects: triples $\left(d : D, Fd \xrightarrow{f} Ge, e : E\right)$
2. Morphisms: arrows $(h, k) : (d, f, e) \to (d', f', e')$ are pairs of D and E-morphisms $h : d \to d', k : e \to e'$ making the following diagram commute:

$$\begin{array}{ccc} Fd & \xrightarrow{Fh} & Fd' \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle f'} \\ Ge & \xrightarrow[Gk]{} & Ge' \end{array}$$

3.1.1 An easy comma: the category of renamings

As a warm-up example, we present the *category of renamings* which can be nicely expressed as an instance of the comma. In elementary terms, the category has the same objects as the syntactic category (contexts, which are finite lists of types) but the morphisms are a restricted class of substitutions, which in simple terms are only allowed to substitute for a variable *by a variable of the same type*. Thus the category of renamings is something like a less proof-relevant version of the category of contexts and substitutions in the following sense: if there exists a substitution relating two contexts in syntactic category, then there exists a renaming relating the same contexts. The upshot is that renamings allow us to track changes in context *due to a substitution* without actually working with substitutions which, because substitutions are identified up to computation, are problematic in a development requiring distinguishing between terms at various stages of reduction (e.g. distinguishing between a term and its normal form.) We will understand this problem in greater detail in Remark 3.2.12. Concisely, the category of renamings allows us to define the presheaves

we need while not losing track of how contexts evolve by substitution[1]. We now present the category abstractly as a comma and at the same time spell out concretely its morphisms in terms of the substitutions of the syntactic category:

The category of renamings has a nice presentation as a comma category.

Definition 3.1.2 (Category of renamings)**.** Recall from Chapter 2 that V is some countably infinite set of variables and $\widetilde{T}$ is the set of all types in the simply typed lambda calculus generated from a set of base types T. Let $\mathbb{F}$ be the category whose objects are finite subsets of V and whose morphisms are all functions between the underlying sets. Let $\mathbb{T} : \mathbb{F} \to \mathfrak{Set}$ be the functor which is constantly $\widetilde{T}$ on objects and constantly $\mathrm{id}_{\widetilde{T}}$ on morphisms. Let $U : \mathbb{F} \to \mathfrak{Set}$ be the forgetful functor taking finite variable sets to their underlying sets and functions to functions. Now by the *category of renamings* we mean the *opposite category* of the comma $U \downarrow \mathbb{T}$ whose

- Objects are *pairs* $\left(V : \mathbb{F}, \Gamma : V \to \widetilde{T}\right)$, i.e., finite variable list together with an assignments of types to a each variable. Note that we elide the final entry of the triple of the comma objects, since it adds no extra information. These objects are precisely the contexts of the familiar classifying category.
- Morphisms of contexts $(V, \Gamma) \to (V', \Gamma')$ are functions $\rho : V \to V'$ making the following diagram commute:

$$\begin{array}{ccc} V & \xrightarrow{\rho} & V' \\ {\scriptstyle\Gamma}\downarrow & & \downarrow{\scriptstyle\Gamma'} \\ \widetilde{T} & \xrightarrow[\mathrm{id}_{\widetilde{T}}]{} & \widetilde{T} \end{array}$$

  The reason the comma morphisms are single functions and not pairs is that the second function would not communicate any data, it would just be taken by $\mathbb{T}$ to the identity no matter what it is. We now unpack this means. The functions ρ are *type-preserving renamings*: for each variable $x \in V$, we have that $\Gamma'(\rho x) = \Gamma(x)$. In particular, this restricts ρ to the following classes of morphisms in the classifying category: substitutions of variables for variables of the same type, context extension with new variables, and all compositions of these. The equations are given by the substitution lemma, as in the syntactic category.

We write $\mathrm{Ren}_\Sigma = (U \downarrow \mathbb{T})^{\mathrm{op}}$ for the category of renamings.

Remark 3.1.3. There is an obvious inclusion $\iota : \mathrm{Ren}_\Sigma \to \mathrm{Cn}_\lambda$ which takes contexts to contexts and casts renamings as change-of-variable substitutions. When going

[1]The reader might complain now, as I did, that if we only care about tracking how contexts evolve by substitutionsn then we should simply work with the category of context *weakenings*. The problem here is that the category of weakenings lacks finite products, which precludes the use of a trick (which we will soon see *does* work in the case of the category of renamings) for explicating the binding structure of its category of presheaves

between Ren_Σ and Cn_λ, we will take liberty to implicitly insert this coercion as needed without further warning. It turns out that this inclusion is *faithful* and thus witnesses Ren_Σ as a *subcategory* of the syntactic category.

3.1.2 Variable-arity Kripke relations

In order to motivate the next, example of a comma construction, we will introduce the notion of *variable-arity Kripke relations*. Variable-arity Kripke relations are sort of like relations parameterized by the objects of some category for which inclusion in a relation respects, in a sense to be defined, the morphisms of that category.

Definition 3.1.4 (C-Kripke relation)**.** Let C and S be categories. For a functor $\varsigma : C \to S$, a *C-Kripke relation* R of *arity* ς over an object A of S is a family of sets $\left\{R\left(c\right) \subseteq S\left[\varsigma\left(c\right), A\right]\right\}_{c \in C}$ satisfying the following *monotonicity* condition:

For each morphism $\rho : c' \to c$ in C and every $a : \varsigma\left(c\right) \to A$ in $R\left(c\right)$, the map $a \circ \varsigma\left(\rho\right) : \varsigma\left(c'\right) \to A$ is in $R\left(c'\right)$.

In order to disentangle this admittedly abstract definition, we specialize to the case of Ren_Σ-Kripke relations which are perhaps more graspable.

Example 3.1.5 (A class of Ren_Σ-Kripke relations over Cn_λ)**.** Consider the inclusion functor $\iota : \mathrm{Ren}_\Sigma \to \mathrm{Cn}_\lambda$. Following the definition above, we can define a Ren_Σ-Kripke relation R of *arity* ι over any type[2] τ of Cn_λ by producing a family of sets $\left\{R\left(\Gamma\right) \subseteq \mathrm{Cn}_\lambda\left[\Gamma, \tau\right]\right\}_{\Gamma \in \mathrm{Ren}_\Sigma}$, namely a selection of terms of type τ in each context Γ, along with a proof of the monotonicity condition: for each *renaming* $\rho : \Gamma' \to \Gamma$ in Ren_Σ and every *term* $t : \Gamma \to \tau$ in $R\left(\Gamma\right)$, the map $t \circ \iota\left(\rho\right) : \Gamma' \to \tau$ is in $R\left(\Gamma'\right)$. This example demonstrates that we can think of Kripke relations as families of unary relations for which inclusion in any of the relations in the family, say $R\left(\Gamma\right)$, those unary relations of terms which can be *lifted* along a renaming to yield a proof of inclusion in the relation at any other context Γ' enjoying a renaming into Γ.

Even with a more concrete example, it can be hard to understand what the utility of a construction like this is without knowing how to map between such C-Kripke relations of a given arity. In fact, we will not say explicitly what a C-Kripke morphism is at all. Instead we turn directly to the work of defining the *gluing category* which turns out to generalize C-Kripke relations to *proof relevant logical predicates*. Indeed, there is a a category of C-Kripke relations whose objects are defined as above; that category can be found to be a full subcategory of the gluing category which we introduce next. We refer the reader to [**fiore_semantic_2002**] for more commentary on this result and for a more detailed treatment of Kripke relations in general.

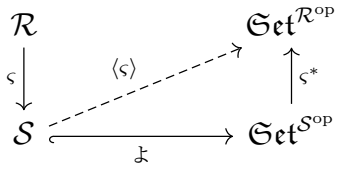
[2] i.e., a singleton context

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

With the category of renamings in hand, we can now introduce as a comma the gluing category which will star as an instrumental semantic domain in our development. To define it, we first need to define a functor along which to take the comma. In a loose sense that will be explained in due course, this functor defines Ren_Σ-presheaves of *open* (semantic) terms.

The relative hom functor

Every functor $\varsigma : \mathcal{R} \to \mathcal{S}$ induces the following situation:



That is, we get a functor

$$\langle\varsigma\rangle : \mathcal{S} \to \mathfrak{Set}^{\mathcal{R}^{\mathrm{op}}}$$

by adjusting the Yoneda embedding into the category of $\mathcal{S}$-presheaves. Specializing to the case of the inclusion $\iota : \mathrm{Ren}_\Sigma \to \mathrm{Cn}_\lambda$ of Remark 3.1.3 gives us a functor

$$\begin{aligned} \mathfrak{Tm} : \mathrm{Cn}_\lambda &\to \widehat{\mathrm{Ren}_\Sigma} \\ \Delta &\mapsto \mathrm{Cn}_\lambda\left(\iota\left(-\right), \Delta\right) \end{aligned}$$

where $\widehat{\mathrm{Ren}_\Sigma}$ is the category $\mathfrak{Set}^{\mathrm{Ren}_\Sigma^{\mathrm{op}}}$. This functor can be construed as taking a syntactic context to a presheaf of open terms. This is a confusing idea at first, and it helps to consider the simple cases to understand it. Consider any singleton context $\tau : \mathrm{Cn}_\lambda$. $\mathfrak{Tm}$ takes τ to the presheaf $\mathrm{Cn}_\lambda\left(-, \tau\right)$ which takes any renaming context Γ to $\mathrm{Cn}_\lambda\left(\Gamma, \tau\right)$. Recalling the definition of the syntactic category and its generating morphisms, the latter set comprises the terms of type τ closed under Γ, represented as single substitutions $\left[t/x\right] : \Gamma \to \tau$. Generalizing to multivariable target contexts Δ gives presheaves of *lists* of open terms of the types in Δ.

It turns out that the category of Ren_Σ presheaves shares the cartesian closed structure of Cn_λ and that $\mathfrak{Tm}$ preserve this structure.

Remark 3.1.6. Any category of presheaves, including $\widehat{\mathrm{Ren}_\Sigma}$, is cartesian closed. Furthermore, the relative hom functor is a *cartesian closed functor*, meaning that for contexts Γ and Δ we have the following isomorphisms of presheaves:

1. $\mathfrak{Tm}\left(\Gamma \times \Delta\right) \cong \mathfrak{Tm}\left(\Gamma\right) \times \mathfrak{Tm}\left(\Delta\right)$
2. $\mathfrak{Tm}\left(\Gamma^\Delta\right) \cong \mathfrak{Tm}\left(\Gamma\right)^{\mathfrak{Tm}(\Delta)}$

The reader should consult Chapter 8 of Awodey's *Category Theory* [**awodey_category_2010**] for more on these results.

Gluing syntax to semantics along the relative hom functor

We at last define the *gluing category* as the comma of the category of renamings along the relative hom functor $\mathfrak{Tm}$ just defined.

Definition 3.1.7 (The gluing category)**.** The gluing category $\mathrm{Gl}_{\mathcal{L}}$ is defined as the comma $\widehat{\mathrm{Ren}_\Sigma} \downarrow \mathfrak{Tm}$. Explicitly, its objects are triples $\left(R : \widehat{\mathrm{Ren}_\Sigma}, q : R \Rightarrow \mathfrak{Tm}\left(\Delta\right), \Delta : \mathrm{Cn}_\lambda\right)$. The objects of the gluing category are *proof relevant logical predicates*, in a sense that will be more clear after a digression to follow. Following the definition of the comma construction, the morphisms $(R, q, \Delta) \to (R', q', \Delta')$ are pairs $\left(d : R \to R', \delta : \mathrm{Cn}_\lambda\left[\Delta', \Delta\right]\right)$ are pairs of a $\widehat{\mathrm{Ren}_\Sigma}$ natural transformation and a substitution making the following diagram commute:

$$\begin{array}{ccc} R & \xrightarrow{\quad d \quad} & R' \\ {\scriptstyle q}\downarrow & & \downarrow{\scriptstyle q'} \\ \mathfrak{Tm}\left(\Delta\right) & \xrightarrow{\mathfrak{Tm}(\delta)} & \mathfrak{Tm}\left(\Delta'\right) \end{array}$$

We can get a foothold understanding of the objects of the gluing category by considering them as a presheaf together with, for each Γ, an interpretation q_Γ of the sets of the presheaf into the set of semantic terms $\mathfrak{Tm}\left(\Delta\right)\left(\Gamma\right)$, such that the interpretation commutes with renamings, in a way that will be further explained in just a bit.

Holding off our desire to further understand the objects, we can already understand the morphisms. A morphism (d, δ) is a natural transformation of Ren_Σ-presheaves glued together with a substitution, such that performing on R the transformation d and then looking at the *semantics* of the resultant presheaf R' by interpreting along q' is the same as interpreting the semantics of the original presheaf R along q and then performing a semantic transformation by substituting with δ. This intuition will become more clear when we define gluing objects over *presheaves of syntax* for which the interpretation $q : R \to \mathfrak{Tm}\left(\Delta\right)$ is the actual *interpretation of syntax* in the classifying category.

We can now better understand the objects by understanding how the gluing category subsumes the notion of variable-arity Kripke relations.

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 $\left(D : \widehat{\mathrm{Ren}_\Sigma}, q : D \Longrightarrow \mathfrak{Tm}\left(\Delta\right), \Delta : \mathrm{Cn}_\lambda\right)$ 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 $\tau \in \mathrm{Cn}_\lambda$, we need to find in these data a Ren_Σ-indexed family $\{R_\Gamma\}_{\Gamma \in \mathrm{Ren}_\Sigma}$ of sets of Cn_λ-morphisms into τ subject to the following monotonicity condition: for any renaming $\rho : \Gamma' \to \Gamma$, if $t : \Gamma \to \tau$ is in R_Γ,

then we have that $t \circ \rho : \Gamma' \to \tau$ is in $R_{\Gamma'}$. Looking again to our proposed Kripke predicate objects of the gluing category, we can see that the presheaf R together with the natural mono q define for each Γ a subset of the substitutions $\Gamma \to \tau$. It is tempting to dismiss the naturality of q as bureaucracy, but naturality turns out to be essential to recovering the monotonicity condition. To see this, we need to look at the naturality diagram of q:

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

As mentioned before, because they are monos we may identify the component morphisms with their images. We write $|q_\Gamma| \subseteq \mathfrak{Tm}\left(\Delta\right)\left(\Gamma\right)$ for the image, for each Γ. Now what this diagram says is that for any $q_\Gamma\left(r \in R\left(\Gamma\right)\right) = t \in |q_\Gamma|$, we have that $t \circ \rho = q_{\Gamma'}\left(R\left(\rho\right)\left(r\right)\right) \in |q_{\Gamma'}|$. This is precisely the monotonicity condition for $\mathfrak{C}$-Kripke relations: containment in each predicate is functorial with respect to renaming *up to renaming*. It turns out that the category of Ren_Σ-Kripke relations is a full subcategory of the gluing category $\widehat{\mathrm{Ren}_\Sigma} \downarrow \mathfrak{Tm}$.

The perspective gained above allows for a more conceptual understanding of the objects in the gluing category: the presheaves R define context-indexed families of *witnesses* to the inclusion of terms in the predicate, and q is a quotient map that forgets the difference between distinct witnesses $w, w' \in R\left(\Gamma\right)$ and whose image just records those terms $t \in \mathfrak{Tm}\left(\Delta\right)\left(\Gamma\right)$ taken to be in the predicate; insofar as q is a mono, these objects are exactly variable-arity Kripke relations.

3.2 Presheaves of stratified neutral and normal syntax

In this section, we will enlist a motely crew of *presheaves of syntax* which more-or-less represent families of terms in the lambda calculus. These presheaves are defined over Ren_Σ for reasons that will become apparent later. We will ultimately define lambda algebras over these presheaves, for which we require some understanding of how to represent variables in $\widehat{\mathrm{Ren}_\Sigma}$ and how exponentiation in $\widehat{\mathrm{Ren}_\Sigma}$ corresponds to lambda abstraction. TODO: more discussion: why presheaves of syntax? explain why we even want algebras in the first place — it is in order to glue them to the semantic algebra on $\mathfrak{Tm}$

3.2.1 Variables and binding in Renhat

Definition 3.2.1 (Variable presheaf)**.** We can define a presheaf of *typed variables* in $\widehat{\mathrm{Ren}_\Sigma}$ with the Yoneda embedding on Ren_Σ:

$$\mathfrak{V}_\tau = \text{よ}\tau = \mathrm{Ren}_\Sigma\left(-, \tau\right)$$

With that definition, we have $\mathfrak{V}_\tau\left(\Gamma\right) = \mathrm{Ren}_\Sigma\left(\Gamma, \tau\right)$ where the right-hand side is comprised of renamings $\Gamma \to [x : \tau]$ which are *functions* $\rho : \mathrm{dom}\left([x : \tau]\right) \to \mathrm{dom}\left(\Gamma\right)$ such that $\Gamma\left(\rho\left(x\right)\right) = [x : \tau]\left(x\right) = \tau$. That is, the ρ are functions selecting a variable of type τ in Γ. More concisely, we have an isomorphism $\mathfrak{V}_\tau\left(\Gamma\right) \cong \left\{x \mid (x : \tau) \in \Gamma\right\}$ by which we allow ourselves to consider this a *presheaf of syntax*.

The Yoneda lemma delivers the following insight about a special class of exponential objects in $\widehat{\mathrm{Ren}_\Sigma}$. Let $\mathfrak{P} : \widehat{\mathrm{Ren}_\Sigma}$. Exponentiation by a representable/variable presheaf, $\mathfrak{V}_\tau = \text{よ}\tau$ gives

$$\begin{aligned}
\mathfrak{P}^{\mathfrak{V}_\Delta}\left(\Gamma\right) &= \mathfrak{P}^{\text{よ}\Delta}\left(\Gamma\right) \\
&= \widehat{\mathrm{Ren}_\Sigma}\left[\text{よ}\Gamma \times \text{よ}\Delta, \mathfrak{P}\right] && (2) \\
&\cong \widehat{\mathrm{Ren}_\Sigma}\left[\text{よ}\left(\Gamma \times \Delta\right)\right] && \text{(since the Yoneda embedding is cartesian closed)} \\
&\cong \mathfrak{P}\left(\Gamma \times \Delta\right) && \text{(by the Yoneda lemma)}
\end{aligned}$$

Where step (2) follows from the definition of the exponentials in the category of presheaves, c.f. page 46 of [**mac_lane_sheaves_1992**].

The presheaves of typed variables together with the simplified view of variable-exponentiation in $\widehat{\mathrm{Ren}_\Sigma}$ will allow us to more easily define algebras for a new algebraic theory $\mathcal{NN}$ of stratified *neutrals and normals*, which allows for a more fine-grained discussion of normal terms.

3.2.2 Stratifying neutral and normal terms

We introduce a new type system over the syntax and type structure of the lambda calculus as defined in Chapter 2. This new type system will distinguish between *neutral terms*, a special class of *normal terms* which are in some sense beta-redexes whose reduction is *blocked* by a variable in the head term, and all other normal terms. In some sense, *neutral terms* can be regarded as those terms which are normal only because they're "stuck."

Definition 3.2.2 (Neutral and normal judgments)**.**

$$\frac{}{\Gamma \vdash_{\text{ne}} x : \tau}\ (x : \tau) \in \Gamma$$

$$\frac{\Gamma \vdash_{\text{ne}} M : \tau_1 * \tau_2}{\Gamma \vdash_{\text{ne}} \pi_i M : \tau_i} \qquad \frac{\Gamma \vdash_{\text{ne}} t_1 : \tau' \to \tau \qquad \Gamma \vdash_{\text{nf}} t_2 : \tau'}{\Gamma \vdash_{\text{ne}} t_1 t_2 : \tau}$$

$$\Gamma \vdash_{\text{nf}} () : \mathbb{1} \qquad \frac{\Gamma \vdash_{\text{nf}} N_i : \tau_i}{\Gamma \vdash_{\text{nf}} (N_1, N_2) : \tau_1 * \tau_2} \qquad \frac{\Gamma, x : \tau \vdash_{\text{nf}} b : \tau'}{\Gamma \vdash_{\text{nf}} \lambda x : \tau.\, b : \tau \to \tau'}$$

$$\frac{\Gamma \vdash_{\text{ne}} t : \theta}{\Gamma \vdash_{\text{nf}} t : \theta}\ (\theta \in T \text{ A BASE TYPE})$$

Definition 3.2.3 (Algebraic theory of neutrals and normals)**.** The judgments above suggest the definition of a theory with a richer sort structure than that of the theory $\mathcal{L}_{\lambda\alpha\beta\eta}$ defined back in Chapter 2. In particular, the new sort structure will feature both a neutral sort and a normal sort for each type τ, allowing us to define algebras that distinguish between these.

We define $\mathcal{NN}$ as the algebraic theory corresponding (c.f. Theorem 1.2.9) to the category with the following objects and generating morphisms:

1. The objects are the collection generated by taking (syntactic) products and exponentials over the collection $\Sigma = \left\{\mathcal{M}_\tau \mid \tau \in \widetilde{T}\right\} \cup \left\{\mathcal{N}_\tau \mid \tau \in \widetilde{T}\right\} \cup \left\{\mathcal{V}_\tau \mid \tau \in \widetilde{T}\right\}$

2. Generating morphisms, for each $\tau, \tau' \in \widetilde{T}$:

$$\begin{aligned}
\text{var}_\tau &: \mathcal{V}_\tau \to \mathcal{M}_\tau \\
\text{fst}_\tau^{\tau'} &: \mathcal{M}_{\tau \times \tau'} \to \mathcal{M}_\tau \\
\text{snd}_\tau^{\tau'} &: \mathcal{M}_{\tau' \times \tau} \to \mathcal{M}_\tau \\
\text{app}_\tau^{\tau'} &: \mathcal{M}_{\tau' \to \tau} \times \mathcal{N}_{\tau'} \to \mathcal{M}_\tau \\
\text{incl}_\theta &: M_\theta \to N_\theta \\
\text{unit} &: \mathbb{1} \to \mathcal{N}_{\mathbb{1}} \\
\text{pair}_\tau^{\tau'} &: \mathcal{N}_\tau \to \mathcal{N}_{\tau'} \to \mathcal{N}_{\tau \times \tau'} \\
\text{abs}_{\tau \to \tau'} &: {\mathcal{N}_{\tau'}}^{\mathcal{V}_\tau} \to \mathcal{N}_{\tau \to \tau'}
\end{aligned}$$

3. The laws are the usual $\alpha-$, $\beta-$, and $\eta-$rules.

Note that the domain of the unit operation symbol is merely the nullary product of sorts.

Remark 3.2.4. Any $\widehat{\mathrm{Ren}_\Sigma}-$ lambda algebra over the family $\{\mathfrak{X}_\tau\}_{\tau\in\widetilde{T}}$ gives rise to an algebra for the algebraic theory $\mathcal{NN}$ over the family $\{\mathfrak{A}_\tau\}_{\tau\in\widetilde{T}}$ by setting

$$\begin{aligned}\mathfrak{A}_{\mathcal{M}_\tau} &= \mathfrak{X}_\tau\\ \mathfrak{A}_{\mathcal{N}_\tau} &= \mathfrak{X}_\tau\\ \mathfrak{A}_{\mathcal{V}_\tau} &= \mathfrak{V}_\tau\end{aligned}$$

and setting the operation symbols as in the lambda algebra.

Remark 3.2.5 (A lambda algebra of open substitutions)**.** By Theorem 1.5.9, the syntactic category Cn_λ of the $\alpha\beta\eta$-lambda calculus has a lambda algebra, and an induced interpretation of terms $[\![-]\!]$.

The morphisms

$$\begin{aligned}\pi_1 &: [\![\tau]\!]\times[\![\tau']\!]\to[\![\tau]\!]\\ \pi_2 &: [\![\tau]\!]\times[\![\tau']\!]\to[\![\tau']\!]\\ \epsilon &: [\![\tau']\!]^{[\![\tau]\!]}\times[\![\tau]\!]\to[\![\tau']\!]\end{aligned}$$

in the syntactic category can be lifted by $\mathfrak{Tm}$ to $\widehat{\mathrm{Ren}_\Sigma}$ to provide the operations for a lambda algebra over the family $\left\{\mathfrak{Tm}\left(\tau\right)\right\}_{\tau\in\widetilde{T}}$

The operations are given as follows:

$$\begin{aligned}\mathfrak{V}_\tau &\xrightarrow{[\![-]\!]} \mathfrak{Tm}\left(\tau\right)\\ \mathbb{1} &\xrightarrow{i_{\mathbb{1}}} \mathfrak{Tm}\left(\mathbb{1}\right)\\ \mathfrak{Tm}\left(\tau*\tau'\right) &\xrightarrow{\mathfrak{Tm}(\pi_1)} \mathfrak{Tm}\left(\tau\right)\\ \mathfrak{Tm}\left(\tau*\tau'\right) &\xrightarrow{\mathfrak{Tm}(\pi_2)} \mathfrak{Tm}\left(\tau'\right)\\ \mathfrak{Tm}\left(\tau\right)\times\mathfrak{Tm}\left(\tau'\right) &\xrightarrow{i_\pi} \mathfrak{Tm}\left(\tau*\tau'\right)\\ \mathfrak{Tm}\left(\tau'^{\tau}\right)\times\mathfrak{Tm}\left(\tau\right) &\xrightarrow{i_\pi}\mathfrak{Tm}\left(\left(\tau'\right)^{\tau}\times\tau\right) \xrightarrow{\mathfrak{Tm}(\epsilon)} \mathfrak{Tm}\left(\tau'\right)\\ \mathfrak{Tm}\left(\tau'\right)^{\mathfrak{V}_\tau} &\xrightarrow{\cong} \mathfrak{Tm}\left(\left(\tau'\right)^{\tau}\right)\end{aligned}$$

where $i_{\mathbb{1}}$ and i_π are isomorphisms induced by $\mathfrak{Tm}$'s status as a Cartesian closed functor. We write $\mathfrak{Tm}$ for this algebra.

Remark 3.2.6. The lambda algebra just defined induces, by Remark 3.2.4, an $\mathcal{NN}$-algebra over $\mathfrak{Tm}$ with the assignment of sorts

$$\begin{aligned}\mathcal{V}_\tau &\mapsto \mathfrak{V}_\tau\\ \mathcal{M}_\tau &\mapsto \mathfrak{Tm}\left(\tau\right)\\ \mathcal{N}_\tau &\mapsto \mathfrak{Tm}\left(\tau\right)\end{aligned}$$

and letting the operations be exactly those of the lambda algebra: since the $\mathcal{M}_\tau = \mathcal{N}_\tau = \mathfrak{Tm}(\tau)$, the signatures of the required operations are exactly the same. We write $(\mathfrak{Tm}, \mathfrak{Tm})$ for this induced algebra.

3.2.3 Presheaves of syntax over the category of renamings

Definition 3.2.7 (A presheaf of open syntactic terms)**.** For each type $\tau \in \widetilde{T}$, define

$$\mathfrak{L}_\tau = \left\{ t \mid \Gamma \vdash t : \tau \right\}_{\Gamma \in \mathrm{Ren}_\Sigma}$$

Together with the *renaming action*, defined for each $\rho : \Gamma' \to \Gamma$ as

$$\begin{aligned} \rho^* : \left\{ t \mid \Gamma \vdash t : \tau \right\} &\to \left\{ t \mid \Gamma' \vdash t : \tau \right\} \\ t &\mapsto \rho^* t \end{aligned}$$

where $\rho^* t$ is the result of ρ (regarded as a substitution) acting on t by the action of the clone model (c.f. Theorem 1.2.7), the above families in fact define presheaves

$$\mathfrak{L}_\tau : \widehat{\mathrm{Ren}_\Sigma}$$

where functorialty of the renaming action is inherited from that of the clone model as in Theorem 1.2.7.

Remark 3.2.8 (A lambda algebra on the presheaves of open syntax)**.** The presheaves of open syntax $\mathfrak{L}_\tau$ form the objects of an algebra for the theory $\mathcal{L}_{\lambda\alpha\beta\eta}$ defined in Definition 2.3.3.

The operations are given by the usual typing rules for the simply typed lambda calculus, as in Definition 2.1.3. We write $\mathfrak{L}$ for this algebra.

Remark 3.2.9 (A stratified-lambda algebra on the presheaves of open syntax)**.** By Remark 3.2.4, the lambda algebra of open syntax from Definition 3.2.8 induces an algebra of the theory $\mathcal{NN}$.

We can define presheaves of neutral and normal terms similarly, with the same presheaf action by renaming as before.

Definition 3.2.10 (A presheaf of neutral terms)**.** For each $\tau \in \widetilde{T}$, the families

$$\mathfrak{Ne}_\tau = \left\{ t \mid \Gamma \vdash_{\mathrm{ne}} t : \tau \right\}_{\Gamma \in \mathrm{Ren}_\Sigma}$$

define a presheaf

$$\mathfrak{Ne}_\tau : \widehat{\mathrm{Ren}_\Sigma}$$

under the renaming action.

Definition 3.2.11 (A presheaf of normal terms)**.** For each $\tau \in \widetilde{T}$, the families

$$\mathfrak{Nf}_\tau = \left\{ t \mid \Gamma \vdash_{\mathrm{nf}} t : \tau \right\}_{\Gamma \in \mathrm{Ren}_\Sigma}$$

define a presheaf

$$\mathfrak{Nf}_\tau : \widehat{\mathrm{Ren}_\Sigma}$$

under the renaming action.

After teasing out the latent binding structure enjoyed by the category of presheaves, we will find in Definition 3.2.13 that the presheaves of neutrals and normals give rise to a $\mathcal{NN}$-algebra over $\widehat{\mathrm{Ren}_\Sigma}$.

Remark 3.2.12 (Why the syntactic category just won't do)**.** As mentioned earlier, a major motivation for defining the category of renamings in the first place (to which we went to considerable trouble!) is that the syntactic category is an unsuitable base category over which to define presheaves like the ones above. Let's go back to basics and recall that a presheaf is not just a fancy family of sets, but crucially comes with a faithful contravariant action taking a morphism in the base category to a function of sets in the family going in the opposite direction. If we take the example of the presheaf of opens of type τ, $\mathfrak{L}_\tau$, there are no apparent problems. We could define the family $\left\{\mathfrak{L}_\tau(\Delta)\right\}_{\Delta\in\mathrm{Cn}_\lambda}$ essentially as before, and use the following presheaf action:

$$\begin{gathered}\mathrm{Cn}_\lambda[\Delta,\Gamma]\to\mathfrak{Set}\left[\mathfrak{L}_\tau(\Gamma),\mathfrak{L}_\tau(\Delta)\right]\\ \delta\mapsto(t\mapsto\delta^* t)\end{gathered}$$

which acts by actually performing the substitution on a term in the set. This action's utility breaks down however, when considering the presheaves $\mathfrak{Nf}_\tau$. Supposing we were able to define $\mathfrak{Nf}_\tau$ instead over the classifying category, with the same family as before but with the natural substitution action shown above we would have for each substitution $\delta:\Delta\to\Gamma$ a function of sets $\mathfrak{Nf}_\tau(\Gamma)\to\mathfrak{Nf}_\tau(\Delta)$. For a variable $(x:\tau)\in\Gamma$, then, the substitution $\left[(\lambda x:\mathbb{1}.\,x)\,\mathrm{unit}/y\right]$ is taken to a function of sets $\mathfrak{Nf}_\tau(\Gamma,y:\mathbb{1})\to\mathfrak{Nf}_\tau(\Gamma)$ performing that substitution on terms. The term $(\lambda x.\,x)\,\mathrm{unit}$ is evidently not a normal form (since a β-reduction step applies) but x is a normal form in $\mathfrak{Nf}_\tau(\Gamma,y:\mathbb{1})$ so that this action forces $(\lambda x:\mathbb{1}.\,x)\,\mathrm{unit}\in\mathfrak{Nf}_\tau(\Gamma)$. The essence of the problem is that normal forms are not closed under substitutions, so the natural substitution action is ruled out. Whether there is a different suitable action by the classifying category on this family is a good question to ponder.

3.2.4 A stratified lambda-algebra of neutrals and normals in Renhat

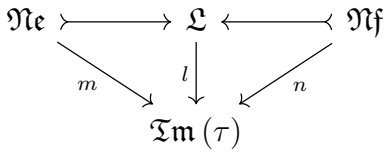
Definition 3.2.13 (A stratified lambda-algebra of neutrals and normals)**.** The presheaves of neutrals and normals defined above give rise to an algebra of the theory $\mathcal{NN}$ in $\widehat{\mathrm{Ren}_\Sigma}$. The sorts $\mathcal{N}_\tau$ (resp. $\mathcal{M}_\tau$) are taken to be the presheaves $\mathfrak{Nf}_\tau$ (resp. $\mathfrak{Ne}_\tau$) and the sorts $\mathcal{V}_\tau$ are taken to be the representable/variable presheaves $\mathfrak{V}_\tau$ defined in Definition 3.2.1.

The operations correspond to the typing rules of Definition 3.2.2.

We write $(\mathfrak{Ne},\mathfrak{Nf})$ for this algebra.

Remark 3.2.14. It turns out that the algebra $(\mathfrak{Ne},\mathfrak{Nf})$ is initial in the category $\mathrm{Mod}_{\widehat{\mathrm{Ren}_\Sigma}}(NN)$ [**fiore_semantic_2002**]. With this fact, the $\mathcal{NN}$-algebra $(\mathfrak{Tm},\mathfrak{Tm})$ over $\left\{\left(\mathfrak{Tm}(\tau),\mathfrak{Tm}(\tau)\right)\right\}_\tau$ together with the $\mathcal{NN}$-algebra over $\left\{(\mathfrak{Ne}_\tau,\mathfrak{Nf}_\tau)\right\}_{\tau\in\widetilde{T}}$ just

presented and the $\mathcal{NN}$-algebra on $\{\mathfrak{L}_\tau\}_{\tau\in\widetilde{T}}$ induce, for each $\tau \in \widetilde{T}$, NN-homomorphisms [3] $l : \mathfrak{L} \to \mathfrak{Tm}$ and $(m, n) : (\mathfrak{Ne}, \mathfrak{Nf}) \to (\mathfrak{Tm}, \mathfrak{Tm})$ such that the following diagram commutes:



In explicit terms, the map l_τ for each $\tau \in \widetilde{T}$ is the usual semantic interpretation of terms in the syntactic category:

$$\begin{aligned} l_\tau\,(\Gamma) : \mathfrak{L}_\tau\,(\Gamma) &\to \mathfrak{Tm}\,(\tau)\,(\Gamma) \\ t &\mapsto [\![\Gamma \vdash t : \tau]\!] \end{aligned}$$

The maps m_τ, n_τ are the usual semantic interpretation precomposed with the respective inclusions into open terms.

[3] in the sense of Definition 1.5.5

Chapter 4

Obtaining a normalization function

4.1 Desiderata for a normalization function

For all this talk of normals forms and normalization, we haven't said much about what a normalization function must be. Of course, it should take a lambda term to a normal form, but the desiderata are far more than that. The following are some of the important properties a normalization function $\mathrm{nf}_\tau^\Gamma : \mathfrak{L}_\tau(\Gamma) \to \mathfrak{Nf}_\tau(\Gamma)$ should satisfy:

- Semantics preservation: For all terms $t \in \mathfrak{L}_\tau(\Gamma)$,

$$\mathrm{nf}_\tau^\Gamma(t) \equiv_{\beta\eta} t$$

- Equation preservation: For all terms $t, t' \in \mathfrak{L}_\tau(\Gamma)$,

$$t \equiv_{\beta\eta} t' \Rightarrow \mathrm{nf}_\tau^\Gamma(t) = \mathrm{nf}_\tau^\Gamma\left(t'\right)$$

These properties together with the signature of the function above encode most of what one might reasonably expect of a normalization function, and indeed these are the ones we shall prove about the normalization we plan to construct. The strength of the approach we have built towards so far is that the proofs of the above properties will essentially fall out of having constructed the normalization function as a composite (with the right domain and codomain) of some maps in the gluing category. It should be said that the above are not a complete enumeration of the properties one might request of a normalization function. Some theoreticians might demands that normal forms are *fixed* by the normalization function; that is, $\mathrm{nf}_\tau^\Gamma(N) = N$ for any normal form N. We do not show this; for that, refer the reader to Fiore [**fiore_semantic_2002**].

Now that we understand what a normalization function must be, we can set about constructing it. As hinted at above, our course will be plotted like this:

First, we will elaborate the cartesian closed structure of the gluing category so that we can extend its lambda algebras along that structure to interpretations of lambda terms. Later, we will define an NN-algebra of *glued* neutrals and normals

whose neutral objects μ_θ will serve as the interpretation of a base type θ. Finally, we will define the normalization function as a composite of the interpretation induced by this assignment and (more or less) the some of the glued operations for the NN-algebra mentioned above. It will turn out that the characterizing property of such glued morphisms says exactly that the semantics is preserved by these morphisms, from which the first property is almost trivial. The second property is even more immediate, as well will find. [TODO: do another pass of this intro]

Definition 4.1.1 (Forgetful projections)**.** The assignments

$$\begin{aligned} \mathrm{Gl}_{\mathcal{L}} &\to \mathrm{Cn}_\lambda \\ (R, q, \Delta) &\mapsto \Delta \end{aligned}$$

$$\begin{aligned} \mathrm{Gl}_{\mathcal{L}} \left[(R, q, \Delta), \left(R', q', \Delta'\right) \right] &\to \mathrm{Cn}_\lambda \left[\Delta, \Delta'\right] \\ (d, \delta) &\mapsto \delta \end{aligned}$$

form a functor $\pi_{\mathrm{sem}} : \mathrm{Gl}_{\mathcal{L}} \to \mathrm{Cn}_\lambda$ which *forgets the syntax* leaving only the semantic components of a gluing object or gluing morphism.

We similarly get a functor $\pi_{\mathrm{syn}} : \mathrm{Gl}_{\mathcal{L}} \to \widehat{\mathrm{Ren}_\Sigma}$ which strips away the semantics leaving only the syntax defined by

$$\begin{aligned} (R, q, \Delta) &\mapsto R \\ (d, \delta) &\mapsto d \end{aligned}$$

4.2 Cartesian closed structure for the gluing category

In order to give an interpretation of terms in the gluing category, we must give an explicit account of its cartesian closed structure. We will say what its products are, but not prove the universal property: we send the reader to [**sterling_normalization_2018**] for that part. We will however give a new proof of the universal property for the exponentials, which is not shown in either of the sources we consulted on connections between gluing and normalization ([**sterling_normalization_2018**], [**fiore_semantic_2002**]).

Theorem 4.2.1 (Products in the gluing category)**.** The terminal object (nullary product) is $\left(\mathbb{1} : \widehat{\mathrm{Ren}_\Sigma}, t : \mathbb{1} \to \mathfrak{Tm}\left(\mathbb{1}\right), \mathbb{1} : \mathrm{Cn}_\lambda\right)$ where and t is the unique map $\mathbb{1} \xrightarrow{\cong} \mathfrak{Tm}\left(\mathbb{1}\right)$ guaranteed by cartesian closure of $\mathfrak{Tm}$. The binary product $(P, p, \Delta) \times (Q, q, \Gamma)$ of (P, p, Δ) and (Q, q, Γ) is $(P \times Q, r, \Delta \times \Gamma)$ where r is the composite

$$P \times Q \xrightarrow{p \times q} \mathfrak{Tm}\left(\Delta\right) \times \mathfrak{Tm}\left(\Gamma\right) \underset{\cong}{\xrightarrow{i_\pi}} \mathfrak{Tm}\left(\Delta \times \Gamma\right)$$

Theorem 4.2.2 (Exponentials in the gluing category)**.** For objects (R_1, q_1, Δ_1) and (R_2, q_2, Δ_2) in $\mathrm{Gl}_{\mathcal{L}}$, the exponential $(R_2, q_2, \Delta_2)^{(R_1, q_1, \Delta_1)}$ is $\left(R, q, {\Delta_2}^{\Delta_1}\right)$ in the pullback diagram

$$\begin{array}{ccc} R & \xrightarrow{\quad r \quad} & {R_2}^{R_1} \\ {\scriptstyle q}\downarrow & & \downarrow{\scriptstyle q_{2*}} \\ \mathfrak{Tm}\left(\Delta_2^{\Delta_1}\right) & \xrightarrow[q_1{}^* \circ \tilde{p}]{} & \mathfrak{Tm}\left(\Delta_2\right)^{R_1} \end{array}$$

where

$$p = \mathfrak{Tm}\left(\epsilon\right) \circ \left(\mathfrak{Tm}\left(\Delta_2^{\Delta_1}\right) \times \mathfrak{Tm}\left(\Delta_1\right) \xrightarrow{\cong} \mathfrak{Tm}\left(\Delta_2^{\Delta_1} \times \Delta_1\right)\right)$$

is the composition of the lifting of the syntactic category's evaluation to $\widehat{\mathrm{Ren}_\Sigma}$ with the isomorphism witnessing the product preservation of $\mathfrak{Tm}$, and $\tilde{p}$ is its exponential transpose.

Proof. The universal property for exponentials is encoded by the product-hom adjunction. It suffices to show an isomorphism

$$\mathrm{Gl}_{\mathcal{L}}\left[\left(R_X, q_X, \Delta_X\right) \times \left(R_Y, q_Y, \Delta_Y\right), \left(R_Z, q_Z, \Delta_Z\right)\right] \cong \mathrm{Gl}_{\mathcal{L}}\left[\left(R_X, q_X, \Delta_X\right), \left(R_Z, q_Z, \Delta_Z\right)^{\left(R_Y, q_Y, \Delta_Y\right)}\right]$$

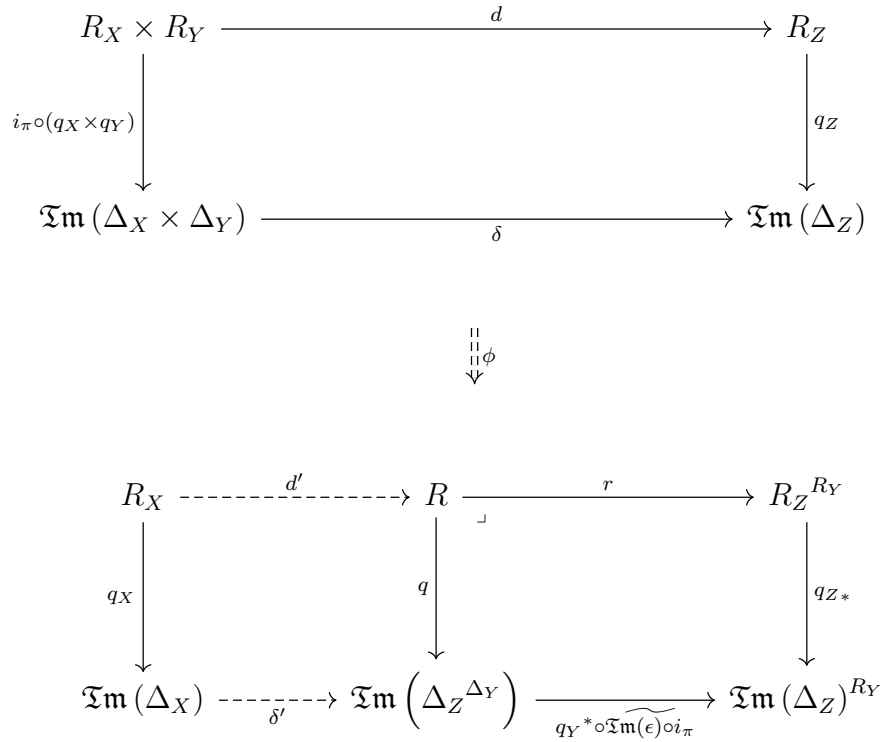
recalling the definition of products in the gluing category, we must show

$$\phi : P \cong E : \psi$$

where

$$\begin{aligned} &\mathfrak{Tm}\left(\Delta_2^{\Delta_1}\right) \times \mathfrak{Tm}\left(\Delta_1\right) \xrightarrow{i_\pi} \mathfrak{Tm}\left(\Delta_2^{\Delta_1} \times \Delta_1\right) \\ &P = \mathrm{Gl}_{\mathcal{L}}\left[\left(R_X \times R_Y, i_\pi \circ \left(q_X \times q_Y\right), \Delta_X \times \Delta_Y\right), \left(R_Z, q_Z, \Delta_Z\right)\right] \\ &E = \mathrm{Gl}_{\mathcal{L}}\left[\left(R_X, q_X, \Delta_X\right), \left(R_Z, q_Z, \Delta_Z\right)^{\left(R_Y, q_Y, \Delta_Y\right)}\right] \end{aligned}$$

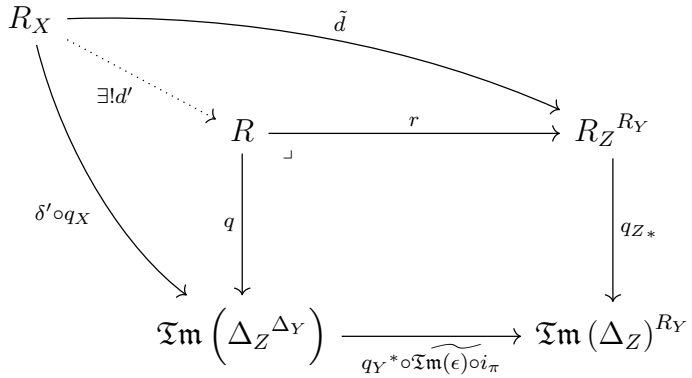
To come up with this isomorphism, it helps to recognize what the morphisms of the relevant gluing objects are. Some consideration reveals that the components of the map ϕ are given by the dotted arrows in the diagram below.



where $\mathfrak{Tm}(\Delta_Z)^{\mathfrak{Tm}(\Delta_Y)} \xrightarrow{i_e} \mathfrak{Tm}\left(\Delta_Z^{\Delta_Y}\right)$ is the isomorphism witnessing exponential preservation for the relative hom functor. Observing the arrows in sight, a clear choice for δ' is

$$i_e \circ \widetilde{\delta \circ i_\pi}$$

The choice for d' is a trickier question. d' should be an arrow into the pullback R. We know nothing about nothing about R except that it fits into the pullback diagram above. The up-side of our limited knowledge of R is that it makes our choice of d' automatic: it must be one of the mediators required by the universal property of the pullback. In particular, we will want to choose $d' =!$ in the diagram below after verifying that the outer square commutes.



To verify that the outer square commutes, We need to demonstrate that

$$q_Y^* \circ \widetilde{\mathfrak{Tm}(\epsilon)} \circ i_\pi \circ i_e \circ \widetilde{\delta \circ i_\pi} \circ q_X = q_{Z*} \circ \tilde{d}$$

which should strike the reader as suspiciously similar to our gluing morphism hypothesis that

$$\delta \circ i_\pi \circ (q_X \times q_Y) = q_Z \circ d$$

Passing to our higher order notation in $\widehat{\mathrm{Ren}_\Sigma}$, we can express the left-hand side of the desired equality as a "lambda expression" (TODO phrasing)

$$\begin{aligned} &\lambda y'.\ \left(\lambda y.\ \delta\left(\left(q_X\left(x : R_x, y\right)\right)\right)\right) q_Y\left(y'\right) \\ =&\lambda y.\ \delta\left(q_X\left(x : R_X\right), q_Y\left(y\right)\right) && (\alpha, \beta) \end{aligned}$$

The transpose of the lower composite in the original gluing morphism diagram is $\widetilde{\delta \circ i_\pi \circ (q_X \times q_Y)}$; it can be rendered as a lambda expression as $\lambda y : R_Y.\ \delta\left(q_X \times q_Y\left(x : R_X, y\right)\right)$ hence by the definition of the product functor, the left-hand side of our desired equality is precisely the transpose of the lower composite in the original gluing morphism diagram.

The right-hand side of the desired equality is the transpose of the upper composite in the gluing morphism diagram so that the desired equality holds because transposition preserves equality of morphisms (TODO: cite? this is obviously true by an appeal to the lambda calculus, but I'm not sure whether this is a well-known theorem in cat theory.)

We now summarize the rightward map for the isomorphism:

$$\begin{aligned} &\phi_1\left(d\right) = d' && \text{(the unique mediating arrow from the argument above)} \\ &\phi_2\left(\delta\right) = i_e \circ \widetilde{\delta \circ i_\pi} \end{aligned}$$

The leftward side of the isomorphism, ψ, is substantially easier, because we don't need to interact much with the pullback. We define

$$\begin{aligned} \psi_1\left(d'\right) &= \epsilon \circ \left(\left(d' \circ r\right) \times \mathrm{id}\right) \\ \psi_2\left(\delta'\right) &= \mathfrak{Tm}\left(\epsilon\right) \circ \left(\delta' \times \mathrm{id}\right) \end{aligned}$$

With our maps in hand, we can verify that they form a bijection:

$$\begin{aligned} \psi_1\left(\phi_1\left(d\right)\right) &= \epsilon \circ \left(\left(d' \circ r\right) \times \mathrm{id}\right) \\ \psi_2\left(\phi_2\left(\delta\right)\right) &= \mathfrak{Tm}\left(\epsilon\right) \circ \left(\left(i_e \circ \widetilde{\delta \circ i_\pi}\right) \times \mathrm{id}\right) \end{aligned}$$

Because we have defined d' by the universal property of the pullback, we have that $d' \circ r = \tilde{d}$ whence $\psi_1\left(\phi_1\left(d\right)\right) = \epsilon \circ \left(\tilde{d} \times \mathrm{id}\right)$ so that $\psi_1\left(\psi_1\left(\,\right)\right) = d$ by the universal property of the transpose in $\widehat{\mathrm{Ren}_\Sigma}$.

The second equality is a little trickier. Both of these are the universal property for the relevant transpose up to squinting, so I think they should be the identity...

TODO:

- finish up the bijection proof
- prove naturality

□

Corollary 4.2.3. The gluing category $\mathrm{Gl}_{\mathcal{L}}$ is cartesian closed.

4.3 Gluing syntax to semantics

We will define some objects in the gluing category which, in some sense, glue the presheaves of syntax we have defined in $\widehat{\mathrm{Ren}_\Sigma}$ together with their semantics in the classifying category. These glued objects will ultimately assemble into the objects for an algebra of the theory NN.

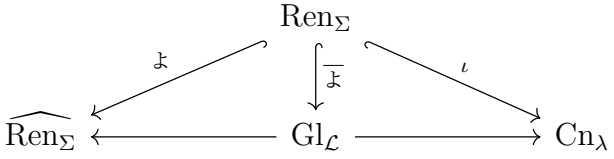
Definition 4.3.1. The relative hom functor induces the embedding

$$\begin{aligned} \overline{よ} : \mathrm{Ren}_\Sigma \quad & \mathrm{Gl}_{\mathcal{L}} \\ \Gamma \mapsto & \left(よ\left(\Gamma\right), よ\left(\Gamma\right) \xrightarrow{\iota_\Gamma} \mathfrak{Tm}\left(\Gamma\right), \Gamma\right) \end{aligned}$$

where

$$\begin{aligned} \left(\underline{\iota}_\Gamma\right)_\Delta : \mathrm{Ren}_\Sigma\left[\Delta, \Gamma\right] &\to \mathrm{Cn}_\lambda\left[\Delta, \Gamma\right] \\ \rho &\mapsto \iota\left(\rho\right) \end{aligned}$$

which fits into the following diagram



$$R \xleftarrow{\pi_{\mathrm{syn}}} (R, q, \Delta) \xrightarrow{\pi_{\mathrm{sem}}} \Delta$$

and satisfies the following form of the Yoneda lemma:

$$(d, \delta) \longmapsto d\,(\mathrm{id})$$

$$\begin{array}{ccc} \mathrm{Gl}_{\mathcal{L}}\left[\overline{\text{ɬ}}\,(-), (R, q, \Delta)\right] & \xrightarrow{\quad\cong\quad} & R\,(-) \\ & {\scriptstyle \pi_{\mathrm{sem}}} \searrow \quad \swarrow {\scriptstyle q} & \\ & \mathrm{Cn}_{\lambda}\left[\iota\,(-), \Delta\right] = \mathfrak{Tm}\,(\Delta) & \end{array}$$

This embedding allows us to define a typed-indexed family of glued objects which glue the syntax, $x, y, z, \ldots$, etc., of variables to the their semantics as substitutions.

Definition 4.3.2 (Gluing syntax and semantics of variables)**.** We define the glued object

$$\nu_{\tau} = \overline{\text{ɬ}}\,(\tau) = \left(\mathfrak{V}_{\tau}, \mathfrak{V}_{\tau} \to \mathfrak{Tm}\,(\tau), \tau\right)$$

where the components of the gluing map are the interpretation of terms in the classifying category.

We do the same for the syntactic presheaves of neutrals and normals we defined earlier. In each case, the components of the quotient map are the interpretation of terms in the classifying category.

Definition 4.3.3 (Gluing syntax and semantics of neutrals)**.**

$$\mu_{\tau} = \left(\mathfrak{Ne}_{\tau}, \mathfrak{Ne}_{\tau} \xrightarrow{m_{\tau}} \mathfrak{Tm}\,(\tau), \tau\right)$$

Definition 4.3.4 (Gluing syntax and semantics of normals)**.**

$$\eta_{\tau} = \left(\mathfrak{Nf}_{\tau}, \mathfrak{Nf}_{\tau} \xrightarrow{n_{\tau}} \mathfrak{Tm}\,(\tau), \tau\right)$$

With these glued objects, we can define an $\mathcal{NN}$-algebra over these families of glued variables, glued neutrals, and glued normals. In turn Definition 1.5.8 will allow us to leverage this to interpret substitutions as gluing category morphisms between these glued objects. The algebra we define will be constructed by gluing together the syntactic $\mathcal{NN}$-algebra in $\widehat{\mathrm{Ren}_{\Sigma}}$ together with the semantic $\mathcal{NN}$-algebra[1] in the classifying category.

Definition 4.3.5 (An algebra of stratified neutrals and normals in the gluing category)**.** The family $\left\{(\mu_{\tau}, \eta_{\tau})\right\}_{\tau \in \widetilde{T}}$ define the objects of an $\mathcal{NN}$-algebra with the operations given as follows:

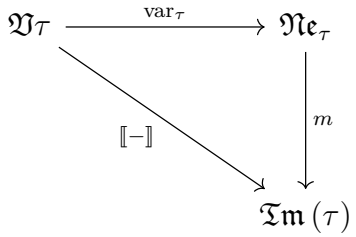
[1] i.e., the one induced by Remark 3.2.4 and the usual lambda algebra in the classifying category

- For each $\tau, \tau' \in \widetilde{T}$, the pair of maps

$$\left(\mathrm{var}_\tau : \mathfrak{V}_\tau \to \mathfrak{Ne}_\tau, \mathrm{id}_{[\![\tau]\!]}\right)$$

is a map $\nu_\tau \to \mu_\tau$ in $\widehat{\mathrm{Ren}_\Sigma} \downarrow \mathfrak{Tm}$.

The gluing property for this pair follows from the fact that $(m, n) : (\mathfrak{Ne}, \mathfrak{Nf}) \to (\mathfrak{Tm}, \mathfrak{Tm})$ is a NN-homomorphism. In particular, we have that the following diagram commutes:



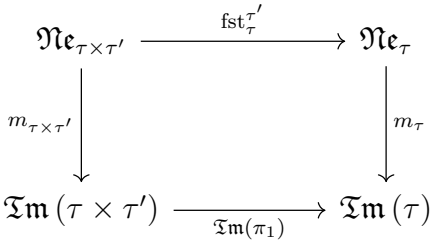
Inserting the lifted identity map at the bottom gives precisely the required square.

- For each $\tau, \tau' \in \widetilde{T}$, the pair of maps

$$\left(\mathrm{fst}_\tau^{\tau'} : \mathfrak{Ne}_{\tau * \tau'} \to \mathfrak{Ne}_\tau, \pi_1 : [\![\tau]\!] \times [\![\tau']\!] \to [\![\tau]\!]\right)$$

is a map $\mu_{\tau * \tau'} \to \mu_\tau$ in $\widehat{\mathrm{Ren}_\Sigma} \downarrow \mathfrak{Tm}$.

The gluing property for this pair again follows from the fact that $(m, n) : (\mathfrak{Ne}, \mathfrak{Nf}) \to (\mathfrak{Tm}, \mathfrak{Tm})$ is a NN-homomorphism. In particular, we have that the following diagram commutes:



Which is exactly the required square. In fact, each of the remaining cases work out in this unsurprising way, so we will stop mentioning it. It is worth noting that one reason these squares line up so cleanly is that we defined the lambda algebra on $\mathfrak{Tm}$ by lifting the relevant operations from Cn_λ into $\widehat{\mathrm{Ren}_\Sigma}$ along $\mathfrak{Tm}$.

- For each $\tau, \tau' \in \widetilde{T}$, the pair of maps

$$\left(\mathrm{snd}_\tau^{\tau'} : \mathfrak{Ne}_{\tau' * \tau} \to \mathfrak{Ne}_\tau, \pi_2 : [\![\tau']\!] \times [\![\tau]\!] \to [\![\tau]\!]\right)$$

is a map $\mu_{\tau' * \tau} \to \mu_\tau$ in $\widehat{\mathrm{Ren}_\Sigma} \downarrow \mathfrak{Tm}$.

- For each $\tau, \tau' \in \widetilde{T}$, the pair of maps

$$\Big(\mathrm{app}_\tau^{\tau'} : \mathfrak{Ne}_{\tau'\to\tau} \times \mathfrak{Nf}_{\tau'} \to \mathfrak{Ne}_\tau, \epsilon : \big([\![\tau']\!] \to [\![\tau]\!]\big) \times [\![\tau']\!] \to [\![\tau]\!]\Big)$$

  is a map $\mu_{\tau'\to\tau} \times \eta_{\tau'} \to \mu_\tau$ in $\widehat{\mathrm{Ren}_\Sigma} \downarrow \mathfrak{Tm}$.

- For each base type $\theta \in T$, the pair of isomorphisms

$$\mathfrak{Ne}_\theta \cong \mathfrak{Nf}_\theta, \mathrm{id}_{[\![\theta]\!]}$$

  is an isomorphism $\mu_\theta \cong \eta_\theta$ in $\widehat{\mathrm{Ren}_\Sigma} \downarrow \mathfrak{Tm}$.

- The pair of isomorphisms

$$\mathbb{1} \cong \mathfrak{Nf}_{\mathbb{1}}, \mathrm{id}_{\mathbb{1}}$$

  is an isomorphism $\mathbb{1} \cong \eta_{\mathbb{1}}$ in $\widehat{\mathrm{Ren}_\Sigma} \downarrow \mathfrak{Tm}$.

- For $\tau, \tau' \in \widetilde{T}$, the pair of isomorphisms

$$\mathrm{pair}_{\tau*\tau'} : \mathfrak{Nf}_\tau \times \mathfrak{Nf}_{\tau'} \xrightarrow{\cong} \mathfrak{Nf}_{\tau*\tau'}, \mathrm{id}_{[\![\tau]\!]\times[\![\tau']\!]}$$

  is an isomorphism $\eta_\tau \times \eta_{\tau'} \xrightarrow{\cong} \eta_{\tau*\tau'}$ in $\widehat{\mathrm{Ren}_\Sigma} \downarrow \mathfrak{Tm}$.

- For $\tau, \tau' \in \widetilde{T}$, the pair of isomorphisms

$$\mathrm{abs}_{\tau\to\tau'} : \mathfrak{Nf}_{\tau'}^{\mathfrak{V}_\tau} \xrightarrow{\cong} \mathfrak{Nf}_{\tau\to\tau'}, \mathrm{id}_{[\![\tau']\!]^{[\![\tau]\!]}}$$

  is an isomorphism $\eta_{\tau'}^{\nu_\tau} \xrightarrow{\cong} \eta_{\tau\to\tau'}$ in $\widehat{\mathrm{Ren}_\Sigma} \downarrow \mathfrak{Tm}$.

Note that the glued operations are given by pairs of syntactic operations and *semantic operations* for elimination forms, and pairs of syntactic operations and the *identity* in the for introduction forms.

Finally we can give an interpretation of base types in the gluing category in terms of the glued neutrals

$$\begin{aligned} T &\xrightarrow{\bar{s}} \widehat{\mathrm{Ren}_\Sigma} \downarrow \mathfrak{Tm} \\ \theta &\mapsto \mu_\theta \end{aligned}$$

Together with a suitable interpretation of base constructors and eliminators (which are taken to be a parameter of this development) as glued morphisms, this interpretation defines a lambda algebra of glued neutrals. Initiality of the classifying category and its lambda algebra in turn induces a unique functor $[\![-]\!]_{\mathrm{Gl}_\mathcal{L}} : \mathrm{Cn}_\lambda \to \mathrm{Gl}_\mathcal{L}$ essentially as in Theorem 1.5.9. We write $[\![-]\!]_{\mathrm{Gl}_\mathcal{L}} : \mathrm{Cn}_\lambda \to \mathrm{Gl}_\mathcal{L}$ for both this functor and the interpretation of terms acquired by precomposing the functor with the usual interpretation of terms in the classifying category. It is crucial to understand that though we have $[\![\theta]\!]_{\mathrm{Gl}_\mathcal{L}} = \mu_\theta$ for base types θ, this equality does *not* hold at higher types

$\tau \in \widetilde{T}$. In the case of products, the interpreted type $[\![\theta \times \theta']\!]_{\mathrm{Gl}_{\mathcal{L}}}$ is the actual product $\mu_\theta \times \mu_{\theta'}$ and not $\mu_{\theta\times\theta'}$. In what follows, we still allow ourselves to write $[\![-]\!]$ for the usual interpretation of terms in the classifying category, distinguishing the new interpretation only by a subscript. The induced interpretation $[\![-]\!]_{\mathrm{Gl}_{\mathcal{L}}}$ extends the usual interpretation of terms in the syntactic category in the following sense: $[\![\Gamma \vdash t : \tau]\!]_{\mathrm{Gl}_{\mathcal{L}}}$ is a pair of the form $\left([\![\Gamma \vdash t : \tau]\!]_{\widehat{\mathrm{Ren}_\Sigma}}, [\![\Gamma \vdash t : \tau]\!]\right)$ where $[\![-]\!]_{\widehat{\mathrm{Ren}_\Sigma}} : \mathrm{Cn}_\lambda \to \widehat{\mathrm{Ren}_\Sigma}$ functor induced by the $\widehat{\mathrm{Ren}_\Sigma}$-lambda algebra over the family $\{\mathfrak{Ne}_\tau\}_{\tau\in\widetilde{T}}$ and initiality of the classifying category.

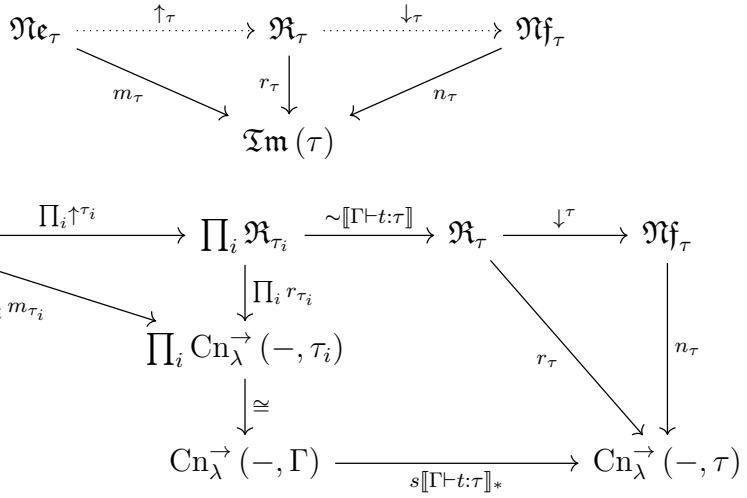
Lemma 4.3.6. The composite $\pi_{\mathrm{sem}} \circ [\![-]\!]_{\mathrm{Gl}_{\mathcal{L}}} : \mathrm{Cn}_\lambda \to \mathrm{Cn}_\lambda$ is the identity endofunctor on Cn_λ.

Proof. This follows from Theorem 1.5.9. In particular, since Cn_λ is the initial category with a model of the lambda calculus, we have that there is a unique functor $\mathrm{Cn}_\lambda \to \mathrm{Cn}_\lambda$ preserving the cartesian closed structure and the lambda algebra. The identity preserves the cartesian closed structure and the lambda algebra, so that uniqueness forces the desired equality. □

Now is a good time look ahead to where we're going, and see how what we've done so far will get us there.

4.4 Taking stock: charting a path to normalization

We now have enough language to say exactly what it is we're hoping to acquire in this chapter. Let $\tau \in \widetilde{T}$ and suppose $[\![\tau]\!]_{\mathrm{Gl}_{\mathcal{L}}} = (\mathfrak{R}_\tau, q, \tau)$. We wish to come up with maps $\uparrow_\tau$ (pronounced "reflect", or for the LISPer "unquote") and $\downarrow_\tau$ (pronounced "reify", or for the LISPer "quote") at each type $\tau \in \widetilde{T}$ to fill in the dotted arrows in the upper diagram such that the lower diagram commutes. In classical approaches to normalization by evaluation, reflection can be construed as lifting syntax into a semantic domain: in practice, this looks like, say, taking syntactic functions in the object language to *actual functions* in the host language. Dually, reification is thought of as lowering semantic objects to a syntactic representation.



Supposing we can achieve this, our normalization function will be precisely the the upper-right composite of the lower diagram. Because the diagram commutes, the normalization function computes for a term t a normal form with the same *semantics* as t; that is, as evident from the diagram, we will have an equality of morphisms $t \equiv \mathrm{nf}\,(t)$ in the syntactic category.

To come up with such maps, we shall produce some intermediate maps

$$\mu_\tau \xrightarrow{\Uparrow^\tau} [\![\tau]\!]_{\mathrm{Gl}_\mathcal{L}} \xrightarrow{\Downarrow^\tau} \eta_\tau$$

in the gluing category. Setting $\downarrow^\tau = \pi_{\mathrm{syn}}\,(\Downarrow^\tau)$ and $\uparrow^\tau = \pi_{\mathrm{syn}}\,(\Uparrow^\tau)$, we find that the triangles above follow from demonstrating that the above gluing maps project in the semantics onto the identity: $\pi_{\mathrm{sem}}\,(\Downarrow^\tau) = \mathrm{id}_{[\![\tau]\!]} = \pi_{\mathrm{sem}}\,(\Uparrow^\tau)$. This follows by considering the universal property of a gluing morphism. In the case of the reflect map, we get a diagram like this

$$\begin{array}{ccc} \mathfrak{Ne}_\tau & \xrightarrow{\pi_{\mathrm{syn}}(\Uparrow^\tau)=\uparrow^\tau} & \mathfrak{R}_\tau \\ {\scriptstyle m_\tau}\downarrow & & \downarrow{\scriptstyle r_\tau} \\ \mathfrak{Tm}\,(\tau) & \xrightarrow[\mathfrak{Tm}(\pi_{\mathrm{sem}}(\Uparrow^\tau))=\mathrm{id}]{} & \mathfrak{Tm}\,(\tau) \end{array}$$

which is the required triangle. In the reification case, we get the desired triangle along the same lines. Convinced thus, we can set about defining the required glued maps.

4.5 Glued reification and reflection

While defining these glued maps as pairs of suitable morphisms, we must take care to verify that the pairs satisfy the gluing condition. Fortunately, the careful setup in the preceding pages makes this process almost automatic.

Definition 4.5.1. The glued maps are defined by induction on the type structure of $\widetilde{T}$:

- For a base type $\theta \in T$, we define $\Uparrow^\theta = \mathrm{id}_{\mu_\theta}$ and $\Downarrow^\theta = \mu_\theta \xrightarrow[\cong]{\mathrm{incl}_\theta} \eta_\theta$
- For the empty product/unit type $\mathbb{1}$, we define $\Uparrow^{\mathbb{1}} = \left(\mu_{\mathbb{1}} \xrightarrow[\cong]{!} \mathbb{1}\right)$ and $\Downarrow^{\mathbb{1}} = \left(\mathbb{1} \xrightarrow[\cong]{(\mathrm{unit},\mathrm{id})} \eta_{\mathbb{1}}\right)$

  The former is the identity which is a map in the gluing category by definition. The latter is an already defined glued map from Definition 4.3.5.
- For types $\tau, \tau' \in \widetilde{T}$, we define

$$\Uparrow^{\tau * \tau'} \colon \mu_{\tau * \tau'} \to [\![\tau]\!]_{\mathrm{Cn}_\lambda} \times [\![\tau']\!]_{\mathrm{Cn}_\lambda}$$

as the pair of the following composites:

$$\mu_{\tau * \tau'} \xrightarrow{\left(\mathrm{fst}^{\tau'}_{\tau}, \pi_1\right)} \mu_\tau \xrightarrow{\Uparrow^\tau} [\![\tau]\!]_{\mathrm{Gl}_\mathcal{L}}$$
$$\mu_{\tau * \tau'} \xrightarrow{\left(\mathrm{snd}^{\tau}_{\tau'}, \pi_2\right)} \mu_{\tau'} \xrightarrow{\Uparrow^{\tau'}} [\![\tau']\!]_{\mathrm{Gl}_\mathcal{L}}$$

and define $\Downarrow^{\tau * \tau'} \colon [\![\tau]\!]_{\mathrm{Gl}_\mathcal{L}} \times [\![\tau']\!]_{\mathrm{Gl}_\mathcal{L}} \to \eta_{\tau * \tau'}$ as the composite

$$[\![\tau]\!]_{\mathrm{Gl}_\mathcal{L}} \times [\![\tau']\!]_{\mathrm{Gl}_\mathcal{L}} \xrightarrow{\Downarrow^\tau \times \Downarrow^{\tau'}} \eta_\tau \times \eta_{\tau'} \xrightarrow[\cong]{(\mathrm{pair}_{\tau * \tau'}, \mathrm{id})} \eta_{\tau * \tau'}$$

In each case, the first pair in the composite is a glued map from Definition 4.3.5 while the second pair is a glued map by the induction hypothesis, so that the composite itself is a glued map. The product of these glued maps is a glued map by the universal property of products.

- For types $\tau, \tau' \in \widetilde{T}$, we define the reflection map

$$\Uparrow^{\tau \to \tau'} \colon \mu_{\tau \to \tau'} \to [\![\tau']\!]_{\mathrm{Gl}_\mathcal{L}}{}^{[\![\tau]\!]_{\mathrm{Gl}_\mathcal{L}}}$$

as the exponential transpose of the composite

$$\mu_{\tau \to \tau'} \times [\![\tau]\!]_{\mathrm{Gl}_\mathcal{L}} \xrightarrow{\mathrm{id} \times \Downarrow^\tau} \mu_{\tau \to \tau'} \times \eta_\tau \xrightarrow{(\mathrm{app}^\tau_{\tau'}, \epsilon)} \mu_{\tau'} \xrightarrow{\Uparrow^{\tau'}} [\![\tau']\!]_{\mathrm{Gl}_\mathcal{L}}$$

By the universal property of the transpose, it suffices to show that the composite is a glued map of the appropriate domain and codomain. This follows from the induction hypothesis, the universal property of products (as in the previous induction step), and since $(\mathrm{app}^\tau_{\tau'}, \epsilon)$ is a glued map from Definition 4.3.5.

We define the reification map as

$$\Downarrow^{\tau \to \tau'} \colon [\![\tau']\!]_{\mathrm{Gl}_\mathcal{L}}{}^{[\![\tau]\!]_{\mathrm{Gl}_\mathcal{L}}} \to \eta_{\tau \to \tau'}$$

as the composite

$$[\![\tau']\!]_{\mathrm{Gl}_\mathcal{L}}{}^{[\![\tau]\!]_{\mathrm{Gl}_\mathcal{L}}} \xrightarrow{\left(\Downarrow^{\tau'}\right)^{\left(\Uparrow^\tau \upsilon_\tau\right)}} \eta_{\tau'}{}^{\nu_\tau} \xrightarrow[\cong]{(\mathrm{abs}_{\tau \to \tau'}, \mathrm{id})} \eta_{\tau \to \tau'}$$

where $\upsilon_\tau = (\mathrm{var}_\tau, \mathrm{id}) : \nu_\tau \to \mu_\tau$.

That this is a glued map follows from Definition 4.3.5 and definition of the post/pre-composition functors TODO: what are these actually called, maybe check out topos theory book?

With the glued maps in hand, we can prove that they do nothing in the semantics. Naturally, the following theorem will prove crucial when demonstrating that our normalization function computes normal forms with the same semantics as the input term. This result is stated but not proven in Fiore's extended abstract [**fiore_semantic_2002**], and the proof is our own.

Theorem 4.5.2 (Reification and reflection are the identity in the semantics)**.** For each type $\tau \in \widetilde{T}$, we have the identities

$$\pi_{\text{sem}}\left(\Uparrow^{\tau}\right) = \text{id}_{\tau} = \pi_{\text{sem}}\left(\Downarrow^{\tau}\right)$$

Proof. The proof proceeds, like the definitions of the reification/reflection maps, by induction on the structure of types:

- For a base type $\theta \in T$, the reflection map is $\Uparrow^{\theta} = \text{id}_{\mu_\theta}$. By functoriality of the interpretation $[\![-]\!]_{\text{Gl}_{\mathcal{L}}} : \text{Cn}_{\lambda} \to \text{Gl}_{\mathcal{L}}$ and Lemma 4.3.6, we have that $\pi_{\text{sem}}\left(\text{id}_{\mu_\theta}\right) = \text{id}_{[\![\theta]\!]_{\text{Cn}_\lambda}}$, as required. The reification map is $\Downarrow^{\theta} = \mu_\theta \xrightarrow[\cong]{\text{incl}_\theta} \eta_\theta$ whose semantic component is the identity by Definition 4.3.5.

- For the empty product/unit type $\mathbb{1}$, the reflection map $\Uparrow^{\mathbb{1}}$ is the unique arrow from $\mu_{\mathbb{1}}$ into the terminal object of the gluing category. The interpretation $[\![-]\!]_{\text{Gl}_{\mathcal{L}}}$ preserves the cartesian closed structure so that $\mu_\theta = \mathbb{1}$ and unique arrow in question is forced to be the identity in $\text{Gl}_{\mathcal{L}}$, whose semantic component is also the identity in Cn_λ.

  The reification map $\Downarrow^{\mathbb{1}}$ is the unit operation for the glued algebra and has the identity as its semantic component by definition.

- For $\tau, \tau' \in \widetilde{T}$, the reflection map $\Uparrow^{\tau\times\tau'}$ for their product is defined as the pair of the following composites:

$$\mu_{\tau * \tau'} \xrightarrow{\left(\text{fst}^{\tau'}_{\tau}, \pi_1\right)} \mu_{\tau} \xrightarrow{\Uparrow^{\tau}} [\![\tau]\!]_{\text{Gl}_{\mathcal{L}}}$$
$$\mu_{tau * \tau'} \xrightarrow{\left(\text{snd}^{\tau}_{\tau'}, \pi_2\right)} \mu_{\tau'} \xrightarrow{\Uparrow^{\tau'}} [\![\tau']\!]_{\text{Gl}_{\mathcal{L}}}$$

  so that

$$\begin{aligned}
\pi_{\text{sem}}\left(\Uparrow^{\tau\times\tau'}\right) &= \left(\pi_{\text{sem}}\left(\Uparrow^{\tau}\right) \circ \pi_1\right) \times \left(\pi_{\text{sem}}\left(\Uparrow^{\tau'}\right) \circ \pi_2\right) \\
&= (\text{id} \circ \pi_1) \times (\text{id} \circ \pi_2) && \text{(by the induction hypothesis)} \\
&= \pi_1 \times \pi_2
\end{aligned}$$

  which is the identity on $[\![\tau]\!]_{\text{Cn}_\lambda} \times [\![\tau']\!]_{\text{Cn}_\lambda}$ as required.

  The reification map $\Downarrow^{\tau*\tau'}$: $[\![\tau]\!]_{\text{Gl}_{\mathcal{L}}} \times [\![\tau']\!]_{\text{Gl}_{\mathcal{L}}} \to \eta_{\tau*\tau'}$ is defined as the composite

$$[\![\tau]\!]_{\text{Gl}_{\mathcal{L}}} \times [\![\tau']\!]_{\text{Gl}_{\mathcal{L}}} \xrightarrow{\Downarrow^{\tau}\times\Downarrow^{\tau'}} \eta_{\tau} \times \eta_{\tau'} \xrightarrow[\cong]{(\text{pair}_{\tau*\tau'}, \text{id})} \eta_{\tau*\tau'}$$

.

whose semantic component is the identity by the induction hypothesis and the definition of the product functor in the gluing category.

- For $\tau, \tau' \in \widetilde{T}$, the reflection map $\Uparrow^{\tau\to\tau'}$ for the function type over these is defined as the exponential transpose of the composite

$$\mu_{\tau\to\tau'} \times [\![\tau]\!]_{\mathrm{Gl}_{\mathcal{L}}} \xrightarrow{\mathrm{id}\times\Downarrow^{\tau}} \mu_{\tau\to\tau'} \times \eta_{\tau} \xrightarrow{(\mathrm{app}^{\tau}_{\tau'},\epsilon)} \mu_{\tau'} \xrightarrow{\Uparrow^{\tau'}} [\![\tau']\!]_{\mathrm{Gl}_{\mathcal{L}}}$$

whence

$$\begin{aligned}
\pi_{\mathrm{sem}}\left(\Uparrow^{\tau\to\tau'}\right) &= \pi_{\mathrm{sem}}\left(\overbrace{\Uparrow^{\tau} \circ(\mathrm{app}^{\tau}_{\tau'},\epsilon)\circ(\mathrm{id}\times\Downarrow^{\tau})}\right)\\
&= \overbrace{\pi_{\mathrm{sem}}\left(\Uparrow^{\tau}\right)\circ\pi_{\mathrm{sem}}\left(\mathrm{app}^{\tau}_{\tau'},\epsilon\right)\circ\pi_{\mathrm{sem}}\left(\mathrm{id}\times\Downarrow^{\tau}\right)} && \text{(by functoriality of } \pi_{\mathrm{sem}})\\
&= \overbrace{\mathrm{id}\circ\epsilon\circ(\mathrm{id}\times\mathrm{id})} && \text{(induction hypothesis, definition of } \pi_{\mathrm{sem}})\\
&= \widetilde{\mathrm{id}\circ\mathrm{id}} && \text{(UP of the transpose, and since } \widetilde{\mathrm{id}} = \mathrm{id})\\
&= \widetilde{\mathrm{id}}\\
&= \mathrm{id}
\end{aligned}$$

as required.

Now the reification map $\Downarrow^{\tau\to\tau'}$ is defined as the composite

$$(\mathrm{abs}_{\tau\to\tau'},\mathrm{id})\circ\left(\Downarrow^{\tau'}\right)^{(\Uparrow^{\tau} v_{\tau})}$$

so that

$$\begin{aligned}
\pi_{\mathrm{sem}}\left(\Downarrow^{\tau\to\tau'}\right) &= \pi_{\mathrm{sem}}\left(\mathrm{abs}_{\tau\to\tau'},\mathrm{id}\right)\circ\left(\pi_{\mathrm{sem}}\Downarrow^{\tau'}\right)^{\pi_{\mathrm{sem}}(\Uparrow^{\tau})\circ\pi_{\mathrm{sem}}(v_{\tau})}\\
&= \mathrm{id}\circ(\mathrm{id})^{\mathrm{id}\circ\mathrm{id}} && \text{(induction hypothesis)}\\
&= \mathrm{id}^{\mathrm{id}}\\
&= \mathrm{id}^{*}\circ\mathrm{id}_{*}
\end{aligned}$$

which is the identity on the exponential $[\![\tau']\!]^{[\![\tau]\!]_{\mathrm{Cn}_{\lambda}}}_{\mathrm{Cn}_{\lambda}}$ in the classifying category.

□

4.6 A promise kept: a function from open terms to normal terms

Having performed reify-reflect yoga at each type, we have established the desired diagram from Section 4.4. We can now define a function $\mathrm{nf}_\tau^\Gamma : \mathfrak{L}_\tau(\Gamma) \to \mathfrak{Nf}_\tau(\Gamma)$ as the composite

$$\mathfrak{L}_\tau(\Gamma) \xrightarrow{l_\tau} \mathfrak{Tm}(\tau)(\Gamma) \xrightarrow{[\![-]\!]} \mathrm{Gl}_{\mathcal{L}}\left[[\![\Gamma]\!], [\![\tau]\!]\right] \xrightarrow{(\Uparrow_\Gamma v_\Gamma)^* \circ \Downarrow^\tau{}_*} \mathrm{Gl}_{\mathcal{L}}\left[\text{ɬ}(\Gamma), \eta_\tau\right] \xrightarrow[\cong]{(d,\delta)\mapsto d(\mathrm{id}_\Gamma)} \mathfrak{Nf}_\tau(\Gamma)$$

where

$$\Uparrow_\Gamma = \prod_{(x:\tau)\in\Gamma} \Uparrow^\tau$$

$$v_\Gamma = \overline{\text{ɬ}}(\Gamma) \xrightarrow{\cong} \prod_{(x:\tau)\in\Gamma} \nu_\tau \xrightarrow{\prod_{(x:\tau)\in\Gamma} v_\tau} \prod_{(x:\tau)\in\Gamma} \mu_\tau$$

recalling that $\nu_\tau \xrightarrow{v_\tau} \mu_\tau$ is the var_τ operation from the $\mathcal{NN}$-algebra on the gluing category.

That is, since the final isomorphism in the composite is a form of the Yoneda lemma and hence given by evaluation at the identity, we have for each term t that $\mathrm{nf}_{\tau,\Gamma}(t) = \left(\Downarrow^\tau [\![\Gamma \vdash t : \tau]\!] \Uparrow_\Gamma v_\Gamma\right)(\mathrm{id}_\Gamma)$

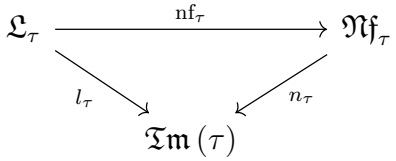
The function we've defined takes open terms to normal forms, but it remains to establish the various correctness properties characterizing a normalization function. The laborious setup work of this chapter turns out to allow us to do so with ease.

4.7 Reaping what we've sown: easy proofs of the correctness properties

Theorem 4.7.1 (Normalization respects computational equality)**.** For every pair of terms $t, t' \in \mathfrak{L}_\tau(\Gamma)$, if $t \equiv_{\beta\eta\alpha} t'$ then $\mathrm{nf}_\tau^\Gamma(t) = \mathrm{nf}_\tau^\Gamma(t')$.

Proof. $\beta\eta\alpha$-equivalent terms have the same interpretation in the classifying category, because the classifying category is quotiented by definitional equality. Symbolically, we have $l_\tau(t) = l_\tau(t')$ from which the required equality is immediate by the definition of the normalization function as a composite starting with l_τ. □

Theorem 4.7.2 (Semantics preservation)**.** The following diagram commutes for every type $\tau \in \widetilde{T}$:



Proof. Per the version of the Yoneda lemma from Definition 4.3.1, we have that the composite $n_\tau \circ \big((d, \delta) \mapsto d\,(\mathrm{id})\big)$ of the gluing map for normal forms with the isomorphism witnessing the Yoneda lemma is the semantic projection π_{sem}. So it suffices to show that $\pi_{\mathrm{sem}}\left(\Downarrow^\tau [\![\Gamma \vdash t : \tau]\!]_{\mathrm{Gl}_\mathcal{L}}\,(\Uparrow_\Gamma v_\Gamma)\right) = [\![\Gamma \vdash t : \tau]\!]_{\mathrm{Cn}_\lambda}$. By Lemma 4.3.6, the interpretation of terms in the gluing category extends the interpretation of terms in the classifying category, so that $\pi_{\mathrm{sem}}\left([\![\Gamma \vdash t : \tau]\!]_{\mathrm{Gl}_\mathcal{L}}\right) = [\![\Gamma \vdash t : \tau]\!]_{\mathrm{Cn}_\lambda}$. Now the desired equality follows from observing that the semantic component of each gluing morphism $\Downarrow^\tau$, $\Uparrow^\tau$, and v_τ each are the identity in the semantic component, by Theorem 4.5.2 and Definition 4.3.5. □

4.8 Looking forward: "blah" by gluing

Conclusion

That's it for now.