Master's Thesis
Logic, Computation and Methodology
Department of Philosophy
Carnegie Mellon University

Impredicative Encodings of Inductive Types in Homotopy Type Theory

Sam Speight

Supervised by:
Prof. Steve Awodey
Dr. Jonas Frey

Submitted:
26th October 2017

Abstract

In Girard's system F, one can give impredicative encodings of various inductive types, such as the coproduct and the type of natural numbers. These encodings satisfy the relevant formation, introduction, elimination and β-computation rules. However, the uniqueness principles (or η-rules) are *not* satisfied. In this project, we move to a more expressive system, Martin-Löf type theory with Σ-types; (intensional) identity types; "large" Π-types over an impredicative universe $\mathcal{U}$ and the function extensionality axiom, to sharpen the impredicative encodings so that the relevant η-rules are satisfied. As a result, the so-determined types have the expected universal properties. Having an impredicative universe allows us to mimic the universal quantification of system F. We first sharpen system F-like encodings of the coproduct and the unit type. We go on to consider a more general method of constructing inductive types: we create initial algebras for endofunctors as limits, using products and equalizers. Impredicativity is crucial for this. The main result is the construction of a type of natural numbers which is the initial algebra for the expected endofunctor $X \mapsto X + \mathbf{1}$.

Acknowledgements

First of all, I would like to thank my advisors, Steve Awodey and Jonas Frey. I would like to thank Steve not only for his expert supervision on this project, but also for making me feel like a valued member of the HoTT group at CMU. I would like to thank Jonas for the large part he played in my category theory education and his unwavering willingness to answer my many questions. It was truly a pleasure to work with both Steve and Jonas, and it was while doing so that I feel that I first took serious strides as a logician—I am very grateful to them for this.

I would also like to thank my family—in particular, my parents, Steve and Jane Speight. It is by no means an exaggeration to say that I would not be where I am without the love and support of my parents. They taught me how to learn and encouraged me to live out my ambitions.

And to my fiancée, Julia: thanks for supporting my decision to hop across the pond and chase my dreams!

Sam Speight
Pittsburgh
Summer 2017

Contents

Chapter 1

Background

1.1 Introduction

Under the Curry-Howard correspondence [How80], Girard's system F is the type-theoretic analog of second-order propositional logic. System F can be obtained as an extension of the simply typed λ-calculus by adding universal quantification (or abstraction) over types (and so is sometimes called the 'polymorphic λ-calculus'). It goes back to Girard [Gir72] and Reynolds [Rey74].

More explicitly, following [SU06], we start with a stock of type variables $(X, Y, Z, ...)$. Then system F types are generated by the following "BNF grammar" [BBG$^+$63].

$$A, B ::= X \mid A \to B \mid \forall X.A$$

Starting with a stock of typed variables $(x, y, z, ...)$, system F *terms* are generated by the following BNF grammar.

$$t, u ::= x \mid \lambda(x : A).t \mid tu \mid \Lambda X.t \mid tA$$

The typing rules for system F are given in Figure 1.1.

On top of these rules we add certain reductions. We have the familiar β-reduction for abstraction-application:

$$(\lambda(x : A).\, t)u \rightsquigarrow_\beta t[u/x]. \tag{1.1.1}$$

But we also add a β-reduction for universal quantification:

$$(\Lambda X.t)A \rightsquigarrow_\beta t[A/X]. \tag{1.1.2}$$

With these reductions, system F enjoys strong normalization [GTL89].

The version of system F we have introduced is the most simple: the only type-formers we have are λ-abstraction and universal quantification. One may wonder about adding further type-formers and, indeed, one may add, for instance, binary

$$\frac{}{\Gamma, x : A \vdash x : A}\ \text{ID}$$

$$\frac{\Gamma, x{:}A \vdash t : B}{\Gamma \vdash \lambda(x : A).\, t : A \to B}\ \to\text{-INTRO} \qquad \frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash tu : B}\ \to\text{-ELIM}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \Lambda X.t : \forall X.A}\ \forall\text{-INTRO}^{*} \qquad \frac{\Gamma \vdash t : \forall X.A}{\Gamma \vdash tB : A[B/X]}\ \forall\text{-ELIM}$$

* where $X \notin \mathrm{FV}(\Gamma)$

Figure 1.1: Typing rules for system F.

products or coproducts—just as in the λ-calculus. However, one of the remarkable things about system F is that these type-formers can be "encoded" using universal quantification. Even data types, such as the natural numbers, can be encoded in system F. With the power of universal abstraction comes great expressiveness. A type encoded using universal quantification is "impredicative," for universal quantification ranges over all types, even the said encoded type.

Let us take coproducts as an example, for we will spend some time on this case in §2.2. In system F, one can define the coproduct of types A and B as follows.

$$A + B := \forall X.(A \to X) \to (B \to X) \to X \tag{1.1.3}$$

This encoding expresses that $A + B$ holds just in case everything holds that follows from A and that also follows from B. Alternatively, it attempts to capture the elimination behavior of the coproduct: we have a map from $A + B$ to any X for which we have maps $A \to X$ and $B \to X$. As we shall see, this encoding fails to satisfy the relevant η-rule and so fails to fully capture the universal property of the coproduct.

The usual rules for the coproduct are given in Figure 1.2. With $A + B$ as in Eq. (1.1.3), we define:

$$\begin{aligned} \mathsf{inl}(a : A) &:= \Lambda X.\lambda(f : A \to X).\, \lambda(g : B \to X).\, fa, \\ \mathsf{inr}(b : B) &:= \Lambda X.\lambda(f : A \to X).\, \lambda(g : B \to X).\, gb, \\ \mathsf{rec}_{+}(s, t)(u) &:= uC(\lambda(x : A).s)(\lambda(y : B).t). \end{aligned} \tag{1.1.4}$$

This is tantamount to the encoding of $A + B$ satisfying the given rules. Even the expected β-rules for the coproduct are satisfied:

$$\begin{aligned} \mathsf{rec}_{+}(s, t)\mathsf{inl}(a) &=_{\beta} s[a/x], \\ \mathsf{rec}_{+}(s, t)\mathsf{inl}(b) &=_{\beta} t[b/y] \end{aligned}$$

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash \mathsf{inl}(a) : A + B}\ {+}\text{-INTRO}_1 \qquad \frac{\Gamma \vdash b : B}{\Gamma \vdash \mathsf{inl}(b) : A + B}\ {+}\text{-INTRO}_2$$

$$\frac{\Gamma, x : A \vdash s : C \qquad \Gamma, y : B \vdash t : C \qquad \Gamma \vdash u : A + B}{\Gamma \vdash \mathsf{rec}_+(s,t)(u) : C}\ {+}\text{-ELIM}$$

Figure 1.2: Inference rules for the coproduct of A and B in system F.

$$\frac{}{\Gamma \vdash 0 : \mathbb{N}}\ \mathbb{N}\text{-INTRO}_1 \qquad \frac{\Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathsf{succ}(n) : \mathbb{N}}\ \mathbb{N}\text{-INTRO}_2$$

$$\frac{\Gamma, x : C \vdash e : C \qquad \Gamma \vdash c : C \qquad \Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathsf{rec}_{\mathbb{N}}(e,c)(n) : C}\ \mathbb{N}\text{-ELIM}$$

Figure 1.3: Inference rules for $\mathbb{N}$ in system F.

(where $=_\beta$ means equal under the two β-reductions (Eq. (1.1.1) and Eq. (1.1.2))).

As already mentioned, we may also encode data types in system F. An example that we will return to in §3.3 is the type of natural numbers. In system F, we can give the following encoding, which, after [Rum04], we refer to as 'Polynat.'

$$\mathbb{N} := \forall X.(X \to X) \to X \to X \tag{1.1.5}$$

We see again that the elimination property is expressed in the encoding: to define a map out of the natural numbers to any type X, it suffices to have an endomap on X as well as a point of X.

As before, the expected rules for $\mathbb{N}$, given in Figure 1.3, are satisfied:

$$\begin{aligned} 0 &:= \Lambda X.\lambda(f : X \to X).\lambda(x : X).x; \\ \mathsf{succ}(n) &:= \Lambda X.\Lambda(f : X \to X).\lambda(x : X).f(nXfx); \\ \mathsf{rec}_{\mathbb{N}}(e,c)(n) &:= nC(\lambda(x : X).e)c, \end{aligned}$$

as well as the β-rules:

$$\begin{aligned} \mathsf{rec}_{\mathbb{N}}(e,c)(0) &=_\beta u \\ \mathsf{rec}_{\mathbb{N}}(e,c)(\mathsf{succ}(n)) &=_\beta s(\mathsf{rec}_{\mathbb{N}}(e,c)(n)) \end{aligned}$$

However, there is a shortcoming with both of these encodings—as well as others, like the product. That is, the well-known encodings do not satisfy the relevant

uniqueness principles (or η-computation rules). The η-rule (though later we will refer to this as the 'weak' η-rule) we would like the coproduct to satisfy is

$$\mathsf{rec}_+(\lambda(x : A).\mathsf{inl}(a), \lambda(y : B).\mathsf{inr}(b))(u) =_\beta u. \tag{1.1.6}$$

For the natural numbers, it is:

$$\mathsf{rec}_{\mathbb{N}}(\mathsf{succ}(n), 0)(n) =_\beta n. \tag{1.1.7}$$

Neither are satisfied. Substituting in the definitions from Eq. (1.1.4) as well as that of $A + B$ from Eq. (1.1.3), the left-hand side of Eq. (1.1.6) becomes:

$$u(\forall X.(A \to X) \to (B \to X) \to X)(\lambda x.\Lambda X.\lambda f.\lambda g.fa)(\lambda y.\Lambda X.\lambda f.\lambda g.gb),$$

which is already in normal form. Likewise for the left-hand side of Eq. (1.1.7):

$$n(\forall X.(X \to X) \to X \to X)(\lambda y.\Lambda X.\Lambda f.\lambda x.f(nXfx))(\Lambda X.\lambda f.\lambda x.x)$$

is in normal form. The types are "too big."

The aim of this project is to build encodings of inductive types, such as the coproduct and the type of natural numbers, that also satisfy the desired uniqueness principles. Actually, as already eluded to, we will show that the encodings of the coproduct and the natural numbers we give satisfy η-rules stronger than those given in Eq. (1.1.6) and Eq. (1.1.7), respectively. As a result of satisfying the relevant η-rules, the so-determined types satisfy the expected universal properties. In order to achieve this, we move from system F to a more expressive system, which we describe in the next section. To emphaize, we build inductive types that satisfy the desired universal properties *internally*; this is in contrast to interpreting a system into a model and having the universal properties hold there—as in the "parametric polymorphism" approach, where the universal quantification of system F is "cut down" on interpretation [Rey83].

1.2 The System

The system in which we shall work can concisely be described as: homotopy type theory (HoTT) with one impredicative universe. That being said, we do no make use of the full power of HoTT; we do not use univalence or any higher inductive types (in fact, the hope is that one could give impredicative encodings of these, too—see Chapter 4). A more precise statement is that we work in (intensional) Martin-Löf type theory with one impredicative universe $\mathcal{U}$ and the function extensionality axiom. More precisely still, we need only postulate Σ-types, intensional identity types and "large" Π-types over an impredicative universe $\mathcal{U}$ ('large' here relates to impredicativity and will be explained). The situation is somewhat similar to that in the calculus of constructions, where one has an impredicative type of *propositions* [CH88].

The function extensionality axiom states that we have a quasi-inverse:

$$\mathsf{funext} : \left(\prod_{x:A} f(x) = g(x)\right) \to f = g$$

to the function (definable by path induction):

$$\mathsf{happly} : f = g \to \prod_{x:A} f(x) = g(x)$$

(for more details, see [Uni13]).

That the universe $\mathcal{U}$ is impredicative gives us large Π-types. This allows us to mimic the universal quantification or abstraction of types in system F. We allow ourselves to form Π over "large" types A, ie. ones that are not in the universe, $\prod_{(x:A)} B(x)$, so long as the type family $B : A \to \mathcal{U}$ is universe-valued. We formulate this in the following rule.

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma, x : A \vdash B : \mathcal{U}}{\Gamma \vdash \prod_{(x:A)} B : \mathcal{U}}\ \Pi\text{-FORM}_1$$

Of course, we also allow ourselves to form dependent functions in the usual way:

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma, x : A \vdash B\ \mathsf{type}}{\Gamma \vdash \prod_{(x:A)} B\ \mathsf{type}}\ \Pi\text{-FORM}_2$$

$$\frac{\Gamma \vdash A : \mathcal{U} \qquad \Gamma, x : A \vdash B : \mathcal{U}}{\Gamma \vdash \prod_{(x:A)} B : \mathcal{U}}\ \Pi\text{-FORM}_3$$

Although, the last of these rules is derivable, given that we also add the following rule.

$$\frac{\Gamma \vdash A : \mathcal{U}}{\Gamma \vdash A\ \mathsf{type}}\ \mathrm{TP}$$

The full set of inference rules for Π-types is given in Figure 1.4. Furthermore, the rules for Σ-types and identity types are in Figure 1.5 and Figure 1.6, respectively. Many of these rules simply carry over from [Uni13].

Oftentimes, we will want to perform a "restricted quantification". For instance, here we will do lots of quantifying over the type of "small" 0-types (ie. 0-types in $\mathcal{U}$), the subtype of $\mathcal{U}$ which is given by:

$$\mathsf{Set} :\equiv \sum_{X:\mathcal{U}} \mathsf{isSet}(X)\,, \tag{1.2.1}$$

where

$$\mathsf{isSet}(X) :\equiv \prod_{(x,y:X)} \prod_{(p,q:x=y)} p = q\,.$$

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma, x : A \vdash B : \mathcal{U}}{\Gamma \vdash \prod_{(x:A)} B : \mathcal{U}}\ \Pi\text{-FORM}_1$$

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma, x : A \vdash B\ \mathsf{type}}{\Gamma \vdash \prod_{(x:A)} B\ \mathsf{type}}\ \Pi\text{-FORM}_2$$

$$\frac{\Gamma, x{:}A \vdash b : B}{\Gamma \vdash \lambda(x:A).\, b : \prod_{(x:A)} B}\ \Pi\text{-INTRO} \qquad \frac{\Gamma \vdash f : \prod_{(x:A)} B \qquad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B[a/x]}\ \Pi\text{-ELIM}$$

$$\frac{\Gamma, x{:}A \vdash b : B \qquad \Gamma \vdash a : A}{\Gamma \vdash (\lambda(x:A).\, b)(a) \equiv b[a/x] : B[a/x]}\ \Pi\text{-}\beta$$

$$\frac{\Gamma \vdash f : \prod_{(x:A)} B}{\Gamma \vdash f \equiv (\lambda x.\, f(x)) : \prod_{(x:A)} B}\ \Pi\text{-}\eta$$

Figure 1.4: Inference rules for Π-types.

$$\frac{\Gamma \vdash A : \mathcal{U} \qquad \Gamma, x : A \vdash B : \mathcal{U}}{\Gamma \vdash \sum_{(x:A)} B : \mathcal{U}}\ \Sigma\text{-FORM}_1$$

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma, x : A \vdash B\ \mathsf{type}}{\Gamma \vdash \sum_{(x:A)} B\ \mathsf{type}}\ \Sigma\text{-FORM}_2$$

$$\frac{\Gamma, x : A \vdash B\ \mathsf{type} \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : B[a/x]}{\Gamma \vdash (a, b) : \sum_{(x:A)} B}\ \Sigma\text{-INTRO}$$

$$\frac{\begin{array}{c}\Gamma, z : \sum_{(x:A)} B \vdash C\ \mathsf{type} \\ \Gamma, x : A, y : B \vdash g : C[(x,y)/z] \qquad \Gamma \vdash p : \sum_{(x:A)} B\end{array}}{\Gamma \vdash \mathsf{ind}_\Sigma(g, p) : C[p/z]}\ \Sigma\text{-ELIM}$$

$$\frac{\begin{array}{c}\Gamma, z : \sum_{(x:A)} B \vdash C\ \mathsf{type} \qquad \Gamma, x : A, y : B \vdash g : C[(x,y)/z] \\ \Gamma \vdash a : A \qquad \Gamma \vdash b : B[a/x]\end{array}}{\Gamma \vdash \mathsf{ind}_\Sigma(g, (a, b)) \equiv g[a, b/x, y] : C[(a,b)/z]}\ \Sigma\text{-}\beta$$

Figure 1.5: Inference rules for Σ-types.

$$\frac{\Gamma \vdash A : \mathcal{U} \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : A}{\Gamma \vdash a =_A b : \mathcal{U}} =\text{-FORM}_1$$

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : A}{\Gamma \vdash a =_A b\ \mathsf{type}} =\text{-FORM}_2$$

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathsf{refl}_a : a =_A a} =\text{-INTRO}$$

$$\frac{\begin{array}{c}\Gamma, x : A, y : A, p : x =_A y \vdash C\ \mathsf{type} \qquad \Gamma, z{:}A \vdash c : C[z, z, \mathsf{refl}_z / x, y, p] \\ \Gamma \vdash a : A \qquad \Gamma \vdash b : A \qquad \Gamma \vdash q : a =_A b\end{array}}{\Gamma \vdash \mathsf{ind}_=(c, a, b, q) : C[a, b, q / x, y, p]} =\text{-ELIM}$$

$$\frac{\begin{array}{c}\Gamma, x : A, y : A, p : x =_A y \vdash C\ \mathsf{type} \\ \Gamma, z{:}A \vdash c : C[z, z, \mathsf{refl}_z / x, y, p] \qquad \Gamma \vdash a : A\end{array}}{\Gamma \vdash \mathsf{ind}_=(c, a, a, \mathsf{refl}_a) \equiv c[a/z] : C[a, a, \mathsf{refl}_a / x, y, p]} =\text{-}\beta$$

Figure 1.6: Inference rules for identity types.

We refer to 0-types as "sets" as they behave like objects in the category of sets (and we shall often simply use 'sets' to mean *small* sets). We use the following abbreviation.

$$\prod_{X:\mathsf{Set}} B(X) :\equiv \prod_{X:\mathcal{U}} \mathsf{isSet}(X) \to B(X)$$

There is an obvious syntactic translation $(-)^T$ from system F to the system laid out here. For types in context, the general shape of the translation is given by:

$$(X_1, ..., X_n \vdash A)^T := X_1 : \mathcal{U}, ..., X_n : \mathcal{U} \vdash A^T : \mathcal{U},$$

where we have:

$$\begin{aligned} X^T &:= X, \\ (A \to B)^T &:= A^T \to B^T, \\ (\forall X.A)^T &:= \prod_{X:\mathcal{U}} A^T. \end{aligned}$$

The general shape of the translation for terms is

$$(x_1 : A_1, ..., x_n : A_n \vdash t : B)^T := x_1 : {A_1}^T, ..., x_n : {A_n}^T \vdash t^T : A^T,$$

where we have:

$$\begin{aligned} x^T &:= x, \\ (\lambda(x : A).t)^T &:= \lambda(x : A^T).t^T, \\ (tu)^T &:= t^T u^T, \\ (\Lambda X.t)^T &:= \lambda(X : \mathcal{U}).t^T, \\ (tA)^T &:= t^T A^T. \end{aligned}$$

The translation is sound in the sense that if a judgment is derivable in system F, then its translation is derivable also. The translation remains sound if $\mathcal{U}$ is everywhere replaced with Set, or the subtype of $\mathcal{U}$ corresponding to any homotopy-level (h-level). This is because types of a given h-level in $\mathcal{U}$ are closed under Π, given function extensionality. That this holds for a predicative universe is [Uni13, Theorem 7.1.9]. Making the universe impredicative does no harm to the argument there. Rather, we make vital use of the fact that $\mathcal{U}$ is impredicative to ensure that Π over the relevant subtype of $\mathcal{U}$ lands in $\mathcal{U}$.

In order to see that this system is consistent, we present a realizability semantics based on "assemblies" and "modest sets" in Appendix A.

Chapter 2

Sharpening the Encodings

2.1 The Guiding Lemma

As we have already discussed, the usual system F encodings of inductive types like the coproduct do not satisfy the relevant η-computation rules. Of course, the same goes for "system F-like encodings," ie. system F encodings translated into our system as described at the end of §1.2. For example, we have a system F-like encoding of the coproduct of sets A and B in:

$$\prod_{X:\mathsf{Set}} (A \to X) \times (B \to X) \to X, \tag{2.1.1}$$

translating Eq. (1.1.3) and uncurrying.

To see how we might sharpen the encodings in order to fix this, we step back and consider an arbitrary set A. We can make an impredicative, system F-like encoding of A, namely:

$$A' :\equiv \prod_{X:\mathsf{Set}} (A \to X) \to X. \tag{2.1.2}$$

Note that in this project, we will concentrate on giving impredicative encodings of inductive types that are sets (in that they are of the type in Eq. (1.2.1)). This somewhat simplifies matters, as will become clear in proofs that follow. More elaborate methods are needed to deal with types of higher h-level.

Going back to the task in hand, we have a map:

$$\mathsf{eval} :\equiv \lambda(a : A).\lambda(X : \mathsf{Set}).\lambda(f : A \to X).f(a) : A \to A', \tag{2.1.3}$$

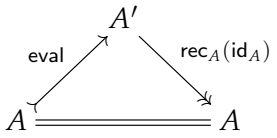
which can be viewed as a constructor for A'. We also have a map:

$$\mathsf{rec}_A(g) :\equiv \lambda(\alpha : A').\alpha_C(g) : A' \to C$$

for any set C and map $g : A \to C$. This can be viewed as an elimination map (or recursor) from A' to C; indeed we would hope to have an elimination map from A' to C whenever we have a map $g : A \to C$.

A special case of the recursor is for the set A and the identity function on A: $\mathsf{rec}_A(\mathsf{id}_A) \equiv \lambda\alpha.\alpha_A(\mathsf{id}_A) : A' \to A$. If A' were a good encoding of A, then we would expect eval and $\mathsf{rec}_A(\mathsf{id}_A)$ to be quasi-inverse: For one, in the presence of univalence, this would make A' propositionally equal to A.

As it happens, the map eval is *not* an equivalence; eval and $\mathsf{rec}_A(\mathsf{id}_A)$ merely form an embedding-retraction pair, ie. the following diagram commutes.



What we have done is motivated the search for a subtype of A', A^*, such that there is an equivalence between A and A^*. We will prove that such an A^* exists, and then go on to use it as the basis for impredicative encodings of some inductive types.

Before doing this, though, let us set up some machinery. Following [Uni13], we define $\mathcal{S}et$ to be the category—in the HoTT sense—whose type of objects is

$$\mathsf{Set} :\equiv \sum_{X:\mathcal{U}} \mathsf{isSet}(X),$$

with

$$\hom_{\mathcal{S}et}(X, Y) :\equiv X \to Y.$$

Note that univalence is required if $\mathcal{S}et$ is to be a category and not just a "precategory." That being said, although we continue to use the terminology of categories, none of our results rely on univalence.

Definition 2.1.4.

For an arbitrary but fixed $A : \mathsf{Set}$, define the **exponentiation functor** $(-)^A : \mathcal{S}et \to \mathcal{S}et$:

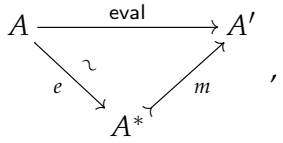
$$X^A :\equiv A \to X$$
$$(f : X \to Y)^A :\equiv f \circ -$$

◇

The following lemma, due to Steve Awodey, will serve as a guide for building some inductive types such as the coproduct (§2.2) and the unit type (§2.4).

Lemma 2.1.5.

For any $A : \mathsf{Set}$*, the map* eval *from Eq.* (2.1.3) *factors as an equivalence* e *followed by an embedding* m*:*



with

$$A^* :\equiv \sum_{(\alpha:A')} \prod_{(X,Y:\mathsf{Set})} \prod_{f:X\to Y} \alpha_Y \circ f^A = f \circ \alpha_X,$$

where

$$A' \equiv \prod_{X:\mathsf{Set}} X^A \to X,$$

which is Eq. (2.1.2) *rewritten using Definition* 2.1.4*.*

Proof. Of course, we define $m :\equiv \mathsf{pr}_1$. Any $\alpha : A'$ is a family of maps $(A \to X) \to X$ for any set X. Then A^* is the subtype of such maps such that the following square commutes.

$$\begin{array}{ccc} X^A & \xrightarrow{\alpha_X} & X \\ {\scriptstyle f^A}\downarrow & & \downarrow{\scriptstyle f} \\ Y^A & \xrightarrow[\alpha_Y]{} & Y \end{array}$$

That is, any $\alpha : A^*$ is a natural transformations from the functor $(-)^A$ to the identity functor $\mathsf{id}_{\mathcal{S}et}$ (justifying the subscript notation).

Observe that A^* is a set: As X and Y are sets, $X^A \to Y$ is a set, given that sets are closed under Π. The type $\alpha_Y \circ f^A = f \circ \alpha_X$ is therefore a proposition. Additionally, A' is a set and sets are closed under Σ, as well.

We define the map e as follows.

$$e(a) :\equiv (\mathsf{eval}(a), p_a : \lambda X.\lambda Y.\lambda f.\mathsf{eval}(a)_Y \circ f^A = f \circ \mathsf{eval}(a)_X)$$

The path p_a can obtained by function extensionality, given the following:

$$\begin{aligned} \mathsf{eval}(a)_Y(f^A(g)) &\equiv \mathsf{eval}(a)_Y(f \circ g) \\ &\equiv (f \circ g)(a) \\ &\equiv f(g(a)) \\ &\equiv f(\mathsf{eval}(a)_X(g)), \end{aligned}$$

for an arbitrary $g : A \to X$.

We now exhibit a quasi-inverse to e.

$$\begin{aligned} e' &: A^* \to A \\ e'(\alpha, q) &:\equiv \alpha_A(\mathsf{id}_A) \end{aligned}$$

Use a_0 to denote $e'(\alpha, q) :\equiv \alpha_A(\mathsf{id}_A)$.

In one direction, we have:

$$e'(e(a)) \equiv e'(\mathsf{eval}(a), p_a) \equiv \mathsf{eval}(a)_A(\mathsf{id}_A) \equiv \mathsf{id}_A(a) \equiv a\,.$$

For the converse direction, we have:

$$e(e'(\alpha, q)) \equiv e(a_0) \equiv (\mathsf{eval}(a_0), p_{a_0})\,.$$

We want to show that $(\mathsf{eval}(a_0), p_{a_0}) = (\alpha, q)$, which is an equality over a Σ-type. So by Lemma 2.7.2 of [Uni13] it is sufficient to show that:

(i) we have a path $r : \mathsf{eval}(a_0) = \alpha$,
(ii) $r_*(p_{a_0}) = q$.

For (i), observe that we have a path:

$$\mathsf{happly}(qAXg, \mathsf{id}_A) : g(\alpha_A(\mathsf{id}_A)) = \alpha_X(g^A(\mathsf{id}_A)).$$

So, using function extensionality we have the required path r, as

$$\mathsf{eval}(a_0)(g) \equiv g(a_0) \equiv g(\alpha_A(\mathsf{id}_A)) = \alpha_X(g^A(\mathsf{id}_A)) \equiv \alpha_X(g \circ \mathsf{id}_A) = \alpha_X(g),$$

for any $g : A \to X$.

The equality in (ii) is over the type $\prod_{(X,Y:\mathsf{Set})} \prod_{(f:X\to Y)} \alpha_Y \circ f^A = f \circ \alpha_X$. But as A^* is a set, this type is a proposition, from which (ii) follows.

□

Observe that the above result can be seen as a special case of the covariant Yoneda Lemma. In category theory:

$$A^* = \mathrm{Hom}_{\mathbf{Set}^{\mathbf{Set}}}(yA, \mathrm{id}_{\mathbf{Set}}) \cong \mathrm{id}_{\mathbf{Set}}(A) = A\,.$$

2.2 The Coproduct

We use Lemma 2.1.5 as the basis for building the coproduct of sets. The rules for the coproduct that we would like to be satisfied are presented in Figure 2.1.

Definition 2.2.1.

Define the **product functor** $(-) \times (-) : \mathcal{S}et \times \mathcal{S}et \to \mathcal{S}et$:

$$A \times B :\equiv A \times B$$
$$((f : X \to Y) \times (g : X' \to Y'))(x : X \times X') :\equiv (f(\mathsf{pr}_1(x)), g(\mathsf{pr}_2(x)))$$

◇

$$\frac{\Gamma \vdash A : \mathsf{Set} \qquad \Gamma \vdash B : \mathsf{Set}}{\Gamma \vdash A + B : \mathsf{Set}}\ +\text{-FORM}$$

$$\frac{}{\Gamma \vdash \mathsf{inl} : A \to A + B}\ +\text{-INTRO}_1 \qquad \frac{}{\Gamma \vdash \mathsf{inr} : B \to A + B}\ +\text{-INTRO}_2$$

$$\frac{\Gamma \vdash C : \mathsf{Set} \qquad \Gamma \vdash s : A \to C \qquad \Gamma \vdash t : B \to C}{\Gamma \vdash \mathsf{rec}_+(s,t) : A + B \to C}\ +\text{-ELIM}$$

$$\frac{\Gamma \vdash C : \mathsf{Set} \qquad \Gamma \vdash s : A \to C \qquad \Gamma \vdash t : B \to C \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathsf{rec}_+(s,t)(\mathsf{inl}(a)) \equiv s(a) : C}\ +\text{-}\beta_1$$

$$\frac{\Gamma \vdash C : \mathsf{Set} \qquad \Gamma \vdash s : A \to C \qquad \Gamma \vdash t : B \to C \qquad \Gamma \vdash b : A}{\Gamma \vdash \mathsf{rec}_+(s,t)(\mathsf{inr}(b)) \equiv t(b) : C}\ +\text{-}\beta_2$$

$$\frac{\begin{array}{c}\Gamma \vdash C : \mathsf{Set} \qquad \Gamma \vdash s : A \to C \qquad \Gamma \vdash t : B \to C \\ f : A + B \to C \qquad \Gamma \vdash p_1 : f \circ \mathsf{inl} = s \qquad \Gamma \vdash p_2 : f \circ \mathsf{inr} = t\end{array}}{\Gamma \vdash p_3 : \mathsf{rec}_+(s,t) = f}\ +\text{-}\eta$$

Figure 2.1: Inference rules for the coproduct.

We trust that context disambiguates between the functor $\times$ and the type-theoretic product $\times$.

So we can now write Eq. (2.1.1) as:

$$(A + B)' :\equiv \prod_{X:\mathsf{Set}} X^A \times X^B \to X\,. \tag{2.2.2}$$

Using Lemma 2.1.5, if $A + B$ is the usual type-theoretic coproduct of sets A and B, then we have:

$$A + B \simeq (A + B)^* :\equiv \sum_{(\alpha:(A+B)')} \prod_{(X,Y:\mathsf{Set})} \prod_{f:X\to Y} \alpha_Y \circ (f^A \times f^B) = f \circ \alpha_X,$$

where $(A + B)'$ is from Eq. (2.2.2). This leads us to make the following definition.

Definition 2.2.3.

Given $A, B : \mathsf{Set}$, we define their **coproduct** as follows:

$$A + B :\equiv \sum_{(\alpha:(A+B)')} \prod_{(X,Y:\mathsf{Set})} \prod_{f:X\to Y} \alpha_Y \circ (f^A \times f^B) = f \circ \alpha_X,$$

where

$$(A + B)' :\equiv \prod_{X:\mathsf{Set}} X^A \times X^B \to X$$

is from Eq. (2.2.2). ◇

In this way, our encoding of $A + B$ is a subtype of the system F-like encoding. But we could not write this encoding in system F, for there we do not have identity types. Observe that $A + B$ here is a set, by reasons discussed in Lemma 2.1.5.

Thus, the usual formation rule for the coproduct of A and B (restricted to sets) is satisfied. Next, we turn to the introduction and elimination rules. We show that the usual introduction rules are satisfied, that is, we define "injections" inl and inr. Moreover, we show that we can eliminate in to other closed sets in the expected way; in other words, the usual simple elimination rule *with respect to other sets* is satisfied.

Lemma 2.2.4.

We can define:

(i) injections $\mathsf{inl} : A \to A + B$ *and* $\mathsf{inr} : B \to A + B$,

(ii) a recursor (or eliminator) $\mathsf{rec}_+(s, t) : A + B \to C$, *for any* $C : \mathsf{Set}$, $s : A \to C$ *and* $t : B \to C$.

Therefore, the introduction rules and the elimination rule from Figure 2.1 are satisfied.

Proof. (i) First we define the two injections.

$$\begin{aligned}\mathsf{inl}(a) :\equiv\ & (\lambda(X:\mathsf{Set}).\lambda(g:(A\to X)\times(B\to X)).\mathsf{pr}_1(g)(a),\\ & \lambda(X:\mathsf{Set}).\lambda(Y:\mathsf{Set}).\lambda(f:X\to Y).\mathsf{refl}_{f\circ\mathsf{pr}_1(g)})\end{aligned}$$

$$\begin{aligned}\mathsf{inr}(b) :\equiv\ & (\lambda(X:\mathsf{Set}).\lambda(g:(A\to X)\times(B\to X)).\mathsf{pr}_2(g)(b),\\ & \lambda(X:\mathsf{Set}).\lambda(Y:\mathsf{Set}).\lambda(f:X\to Y).\mathsf{refl}_{f\circ\mathsf{pr}_2(g)})\end{aligned}$$

(ii) Given any $C:\mathsf{Set}$, $s:A\to C$ and $t:B\to C$, define:

$$\mathsf{rec}_+(s,t) :\equiv \lambda(z:A+B).\mathsf{pr}_1(z)_C(s,t).$$

□

The injections and recursor satisfy the expected β-computation rules definitionally (cf. Figure 2.1).

Lemma 2.2.5 - β-rules for $A+B$.

Let the injections and the recursor be as in Lemma 2.2.4, $C:\mathsf{Set}$, $s:A\to C$ and $t:B\to C$. Then the following hold.

(i) $\mathsf{rec}_+(s,t)(\mathsf{inl}(a)) \equiv s(a)$
(ii) $\mathsf{rec}_+(s,t)(\mathsf{inr}(b)) \equiv t(b)$

Proof. We show (i); (ii) is can be shown similarly.

$$\begin{aligned}\mathsf{rec}_+(s,t)(\mathsf{inl}(a)) &\equiv \lambda z.\mathsf{pr}_1(z)_C(s,t)(\lambda X.\lambda g.\mathsf{pr}_1(g)(a),\lambda X.\lambda Y.\lambda f.\lambda g.\mathsf{refl}_{f\circ\mathsf{pr}_1(g)})\\ &\equiv \mathsf{pr}_1(\lambda X.\lambda g.\mathsf{pr}_1(g)(a),\lambda X.\lambda Y.\lambda f.\lambda g.\mathsf{refl}_{f\circ\mathsf{pr}_1(g)})_C(s,t)\\ &\equiv (\lambda X.\lambda g.\mathsf{pr}_1(g)(a))_C(s,t)\\ &\equiv \mathsf{pr}_1(s,t)(a)\\ &\equiv s(a)\end{aligned}$$

□

It is to be expected that the β-rules hold definitionally, as they are already satisfied by the system F encoding. Recall, the aim is to show that our impredicative encoding of the coproduct also satisfies the η-computation rule. The following is a key lemma towards this aim. It concerns the behavior of the eliminator acting on the constructors; we think of it as a "weak" η-rule for the coproduct.

Lemma 2.2.6 - weak η-rule for $A+B$.

The following propositional equality holds for any $z:A+B$.

$$\mathsf{rec}_+(\mathsf{inl},\mathsf{inr})(z) = z$$

Proof. Unpacking $\mathsf{rec}_+(\mathsf{inl},\mathsf{inr})(z)$, we have to show:

$$\mathsf{pr}_1(z)_{A+B}(\mathsf{inl},\mathsf{inr}) = z. \tag{2.2.7}$$

Here we see the importance of $A+B$ as defined in Definition 2.2.3 being a set, for otherwise we could not apply $\mathsf{pr}_1(z)$ to $A+B$.

As Eq. (2.2.7) is an equality over a Σ-type, we use Lemma 2.7.2 of [Uni13] to show that:

(i) we have a path $p : \mathsf{pr}_1(\mathsf{pr}_1(z)_{A+B}(\mathsf{inl},\mathsf{inr})) = \mathsf{pr}_1(z)$,
(ii) $p_*(\mathsf{pr}_2(\mathsf{pr}_1(z)_{A+B}(\mathsf{inl},\mathsf{inr}))) = \mathsf{pr}_2(z)$.

Using Σ-induction, let

$$z \equiv \left(\alpha : (A+B)' \,,\, q : \prod_{(X,Y:\mathsf{Set})} \prod_{(f:X\to Y)} \alpha_Y \circ (f^A \times f^B) = f \circ \alpha_X \right).$$

Then, for (i), using function extensionality, we want to show:

$$\mathsf{pr}_1(\alpha_{A+B}(\mathsf{inl},\mathsf{inr}))X(f,g) = \alpha_X(f,g), \tag{2.2.8}$$

for arbitrary $f : A \to X$ and $g : B \to X$.

But, looking at the right-hand side of Eq. (2.2.8):

$$\begin{aligned} \alpha_X(f,g) &\equiv \alpha_X(\mathsf{rec}_+(f,g) \circ \mathsf{inl}, \mathsf{rec}_+(f,g) \circ \mathsf{inr}) \\ &\equiv \alpha_X((\mathsf{rec}_+(f,g)^A \times \mathsf{rec}_+(f,g)^B)(\mathsf{inl},\mathsf{inr})), \end{aligned}$$

using Lemma 2.2.5 (the β-rules for the coproduct). On the left-hand side:

$$\begin{aligned} \mathsf{pr}_1(\alpha_{A+B}(\mathsf{inl},\mathsf{inr}))X(f,g) &\equiv \lambda z.\mathsf{pr}_1(z)_X(f,g)(\alpha_{A+B}(\mathsf{inl},\mathsf{inr})) \\ &\equiv \mathsf{rec}_+(f,g)(\alpha_{A+B}(\mathsf{inl},\mathsf{inr})). \end{aligned}$$

That is, Eq. (2.2.8) just expresses the following instance of naturality, evaluated at $(\mathsf{inl},\mathsf{inr})$.

$$\begin{array}{ccc} (A+B)^A \times (A+B)^B & \xrightarrow{\alpha_{A+B}} & A+B \\ \downarrow{\scriptstyle \mathsf{rec}_+(f,g)^A \times \mathsf{rec}_+(f,g)^B} & & \downarrow{\scriptstyle \mathsf{rec}_+(f,g)} \\ X^A \times X^B & \xrightarrow[\alpha_X]{} & X \end{array}$$

But, of course, we can get an inhabitant of the type of Eq. (2.2.8), namely:

$$(\mathsf{happly}(q(A+B)X\mathsf{rec}_+(f,g),(\mathsf{inl},\mathsf{inr})))^{-1}.$$

The equality in (ii) is over the type:

$$\prod_{(X,Y:\mathsf{Set})} \prod_{(f:X\to Y)} \alpha_Y \circ (f^A \times f^B) = f \circ \alpha_X.$$

But as Y is a set, this type is a proposition, from which (ii) follows.

□

Now we come to the desired theorem, that the η-computation rule for the coproduct is satisfied by our impredicative encoding. We call this result the η-rule for it expresses the uniqueness clause of the universal property of the coproduct—as categorically-minded readers will recognize. Moreover, in this way, it is a special case of the η-rule for W-types as formulated in [AGS12].

Theorem 2.2.9 - η-rule for $A + B$.

Recall the $+$-η rule from Figure 2.1. Let $C : \mathsf{Set}$, $s : A \to C$, $t : B \to C$ and $f : A + B \to C$ such that

$$f \circ \mathsf{inl} = s,$$
$$f \circ \mathsf{inr} = t.$$

Then

$$\mathsf{rec}_+(s,t) = f.$$

Proof. We show that:

$$\mathsf{rec}_+(s,t) = \mathsf{rec}_+(f \circ \mathsf{inl}, f \circ \mathsf{inr}) = f \circ \mathsf{rec}_+(\mathsf{inl}, \mathsf{inr}) = f. \tag{2.2.10}$$

The first of the equalities in Eq. (2.2.10) follows by unpacking the definition of the recursor and using function extensionality, given the hypotheses of the theorem. The last equality follows from Lemma 2.2.6 and function extensionality. So we are just left to show that the middle equality holds.

Unpacking the definition of the recursor we have:

$$\mathsf{rec}_+(f \circ \mathsf{inl}, f \circ \mathsf{inr}) \equiv \lambda z.\mathsf{pr}_1(z)_C(f \circ \mathsf{inl}, f \circ \mathsf{inr}).$$
$$f \circ \mathsf{rec}_+(\mathsf{inl}, \mathsf{inr}) \equiv f \circ \lambda z.\mathsf{pr}_1(z)_{A+B}(\mathsf{inl}, \mathsf{inr}).$$

So, using function extensionality and Σ-induction to give each of the above functions an argument

$$z \equiv \left(\alpha : (A+B)', p : \prod_{(X,Y:\mathsf{Set})} \prod_{(f:X\to Y)} \alpha_Y \circ (f^A \times f^B) = f \circ \alpha_X \right),$$

it suffices to show:

$$\alpha_C(f \circ \mathsf{inl}, f \circ \mathsf{inr}) = f(\alpha_{A+B}(\mathsf{inl}, \mathsf{inr})).$$

For this, the path p holds the key;

$$\mathsf{happly}(p(A+B)Cf, (\mathsf{inl}, \mathsf{inr}))$$

is of the required type.

□

In summary, we have given an impredicative encoding of the coproduct—as a subtype of the system F-like encoding—which has the expected universal property (in the category $\mathcal{S}et$).

2.3 Induction

As we are working in a dependent type theory, what we would really like is to have our impredicative encodings satisfy the relevant *dependent* elimination rules (including β- and η-computation rules—altogether, induction principle). In [AGS12], Awodey; Gambino and Sojakova show that the following are equivalent for W-types:

(R) the recursion principle: the simple elimination rule together with *propositional* β- and η-rules.

(I) the induction principle: dependent elimination rule together with *propositional* β-rule.

A propositional η-rule for the dependent eliminator follows from the dependent elimination rule. Of course, (I) $\Rightarrow$ (R) follows by taking any constant type family. Given the results of the previous section, this equivalence applies to our impredicative encoding of the coproduct from the previous section. We give some details towards the fact that (R) $\Rightarrow$ (I) in the case of the coproduct.

Theorem 2.3.1.

Suppose $A + B$ satisfies the recursion principle for the coproduct of A and B (the simple elimination rule and β- and η-rules from Figure 2.1). Then $A + B$ satisfies the dependent elimination rule for the coproduct, given by:

$$\frac{\Gamma, z : A+B \vdash C(z) : \mathsf{Set} \qquad \Gamma \vdash s(a) : C(\mathsf{inl}(a)) \qquad \Gamma \vdash t(b) : C(\mathsf{inr}(b)) :}{\Gamma, z : A+B \vdash \mathsf{ind}_{A+B}(s,t)(z) : C(z)} \; +\text{-IND} \; .$$

Proof. Assume the recursion principle for the coproduct of A and B. Assume also the premises of the $+$-IND rule, ie. assume $z : A+B \vdash C(z) : \mathsf{Set}$ and $s(a) : C(\mathsf{inl}(a))$ and $t(b) : C(\mathsf{inr}(b))$. We want to show the conclusion of the rule, ie. $z : A+B \vdash$

$\mathsf{ind}_+(s,t)(z) : C(z)$. Consider the following.

$$\begin{aligned} X &:\equiv \sum_{z:A+B} C(z) \\ i &: \lambda a.(\mathsf{inl}(a), s(a)) : A \to X \\ j &: \lambda b.(\mathsf{inr}(b), t(b)) : B \to X \end{aligned}$$

Since X is a closed type, by the simple elimination rule, we obtain a map $\mathsf{rec}_+(i,j) : A + B \to X$. Moreover, we have that:

$$\begin{aligned} \mathsf{rec}_+(i,j)(\mathsf{inl}(a)) &= i(a), \\ \mathsf{rec}_+(i,j)(\mathsf{inr}(b)) &= j(b), \end{aligned}$$

by the β-rules.

Now consider the map $\mathsf{pr}_1 : X \to A + B$. Observe that we have:

$$\begin{aligned} \mathsf{pr}_1(i(a)) &= \mathsf{inl}(a), \\ \mathsf{pr}_1(j(b)) &= \mathsf{inr}(b). \end{aligned}$$

So:

$$\mathsf{pr}_1 \circ \mathsf{rec}_+(i,j) : A + B \to A + B$$

and, moreover:

$$\begin{aligned} \mathsf{pr}_1 \circ \mathsf{rec}_+(i,j) \circ \mathsf{inl} &= \mathsf{inl}, \\ \mathsf{pr}_1 \circ \mathsf{rec}_+(i,j) \circ \mathsf{inr} &= \mathsf{inr}. \end{aligned}$$

by function extensionality. But this means that we have a path

$$p : \mathsf{pr}_1 \circ \mathsf{rec}_+(i,j) = \mathsf{rec}_+(\mathsf{inl}, \mathsf{inr}),$$

by the η-rule for the coproduct $A + B$.

This allows us to transport using $p_* : C(\mathsf{pr}_1(\mathsf{rec}_+(i,j)(z))) \to C(z)$, so that we may define:

$$\mathsf{ind}_+(s,t)(z) :\equiv (\mathsf{happly}(p,z))_*(\mathsf{pr}_2(\mathsf{rec}_+(i,j)(z))) : C(z).$$

The transport here is necessary, for in general it is not the case that, for example,

$$\mathsf{pr}_1(\mathsf{rec}_+(i,j)(\mathsf{inl}(a))) \equiv \mathsf{inl}(a),$$

and so $\mathsf{pr}_2(\mathsf{rec}_+(i,j)(\mathsf{inl}(a)))$ need not be of type

$$C(\mathsf{inl}(a)) \equiv C(\mathsf{pr}_1(\mathsf{rec}_+(i,j)(\mathsf{inl}(a)))),$$

as it is required to be.

□

$$\frac{}{\Gamma \vdash \mathbf{1} : \mathsf{Set}}\ \mathbf{1}\text{-FORM} \qquad \frac{}{\Gamma \vdash \star : \mathbf{1}}\ \mathbf{1}\text{-INTRO} \qquad \frac{\Gamma \vdash C : \mathsf{Set} \qquad \Gamma \vdash c : C}{\Gamma \vdash \mathsf{rec}_{\mathbf{1}}(c) : \mathbf{1} \to C}\ \mathbf{1}\text{-ELIM}$$

$$\frac{\Gamma \vdash C : \mathsf{Set} \qquad \Gamma \vdash c : C}{\Gamma \vdash \mathsf{rec}_{\mathbf{1}}(c)(\star) \equiv c : C}\ \mathbf{1}\text{-}\beta \qquad \frac{\Gamma \vdash C : \mathsf{Set} \qquad \Gamma \vdash f : \mathbf{1} \to C}{\Gamma \vdash p : \mathsf{rec}_{\mathbf{1}}(f(\star)) = f}\ \mathbf{1}\text{-}\eta$$

Figure 2.2: Inference rules for the unit type.

Notice that in the preceding proof, we only use a special case of the η-rule as formulated in Theorem 2.2.9, the case given by:

$$\mathsf{rec}_{+}(s,t) = f$$

for any

$$s : A \to A + B,$$
$$t : A \to A + B$$

and

$$f : A + B \to A + B$$

such that

$$f \circ \mathsf{inl} = s,$$
$$f \circ \mathsf{inr} = t.$$

That is, the equivalence between the simple elimination rule and the dependent elimination rule holds in the presence of this somewhat weaker η-rule (though not the "weak" η-rule of Lemma 2.2.6). Looking over the details of [AGS12], we see that this observation generalizes to the case of W-types.

2.4 The Unit Type

We shall give one more impredicative encoding based on Lemma 2.1.5, that of the unit type. The rules that we would like to be satisfied are presented in Figure 2.2.

Definition 2.4.1.

We define the **unit type** as follows:

$$\mathbf{1} :\equiv \sum_{(\alpha:\mathbf{1}')} \prod_{(X,Y:\mathsf{Set})} \prod_{f:X\to Y} \alpha_Y \circ f = f \circ \alpha_X,$$

where

$$\mathbf{1}' :\equiv \prod_{X:\mathsf{Set}} X \to X.$$

$\diamond$

The sole constructor and the recursor for the unit type are defined in the next lemma, which mean that the usual introduction rule and simple elimination rule with respect to other sets for $\mathbf{1}$ are satisfied.

Lemma 2.4.2.

We can define:

(i) $\star : \mathbf{1}$,
(ii) a recursor $\mathsf{rec}_{\mathbf{1}}(c) : \mathbf{1} \to C$, *for any* $C : \mathsf{Set}$ *and* $c : C$.

Proof. (i) $\star :\equiv (\lambda X.\mathsf{id}_X, \lambda X.\lambda Y.\lambda f.\mathsf{refl}_f)$
(ii) $\mathsf{rec}_{\mathbf{1}}(c) :\equiv \lambda z.\mathsf{pr}_1(z)_C(c)$

□

The β-rule again holds *definitionally*.

Lemma 2.4.3 - β-rule for 1.

Let $C : \mathsf{Set}$ *and* $c : C$. *Then the following holds.*

$$\mathsf{rec}_{\mathbf{1}}(c)(\star) \equiv c$$

Proof.

$$\begin{aligned}\mathsf{rec}_{\mathbf{1}}(c)(\star) &\equiv \lambda z.\mathsf{pr}_1(z)_C(c)(\lambda X.\mathsf{id}_X, \lambda X.\lambda Y.\lambda f.\mathsf{refl}_f)\\ &\equiv \mathsf{pr}_1(\lambda X.\mathsf{id}_X, \lambda X.\lambda Y.\lambda f.\mathsf{refl}_f)_C(c)\\ &\equiv (\lambda X.\mathsf{id}_X)_C(c)\\ &\equiv \mathsf{id}_C(c)\\ &\equiv c\end{aligned}$$

□

Lemma 2.4.4 - weak η-rule for 1.

The following propositional equality holds for any $z : \mathbf{1}$.

$$\mathsf{rec}_{\mathbf{1}}(\star)(z) = z$$

Proof. Using the definition of the recursor and function extensionality, it suffices to show:

$$\mathsf{pr}_1(z)_{\mathbf{1}}(\star) = z\,.$$

By Σ-induction, we let $z \equiv (\alpha : \mathbf{1}', p : \prod_{(X,Y:\mathsf{Set})} \prod_{(f:X\to Y)} \alpha_Y \circ f = f \circ \alpha_X)$. Then we want to show $\alpha_{\mathbf{1}}(\star) = (\alpha, p)$. As this equality is over a Σ-type, it suffices to show:

(i) we have a path $q : \mathsf{pr}_1(\alpha_{\mathbf{1}}(\star)) = \alpha$,

(ii) $q_*(\mathsf{pr}_2(\alpha_{\mathbf{1}}(\star)) = p$.

For (i), by function extensionality, it suffices to show:

$$\mathsf{pr}_1(\alpha_{\mathbf{1}}(\star))Xx = \alpha_X(x). \tag{2.4.5}$$

But this just expresses the following naturality square evaluated at $\star$.

$$\begin{array}{ccc} \mathbf{1} & \xrightarrow{\alpha_{\mathbf{1}}} & \mathbf{1} \\ {\scriptstyle \mathsf{rec}_{\mathbf{1}}(x)}\downarrow & & \downarrow{\scriptstyle \mathsf{rec}_{\mathbf{1}}(x)} \\ X & \xrightarrow[\alpha_X]{} & X \end{array},$$

for $\alpha_X(\mathsf{rec}_{\mathbf{1}}(x)(\star)) \equiv \alpha_X(x)$ by the β-rule for $\mathbf{1}$ (Lemma 2.4.3). The term

$$(\mathsf{happly}(p\mathbf{1}X\mathsf{rec}_{\mathbf{1}}(x), \star))^{-1}$$

inhabits the type in Eq. (2.4.5).

(ii) follows from the fact that Y is a set.

□

This brings us to the η-rule for $\mathbf{1}$.

Theorem 2.4.6 - η-rule for 1.

If $C : \mathsf{Set}$ *and* $f : \mathbf{1} \to C$, *then*

$$\mathsf{rec}_{\mathbf{1}}(f(\star)) = f.$$

Proof. By Lemma 2.4.4, $f = f \circ \mathsf{rec}_{\mathbf{1}}(\star)$, so, by function extensionality, it suffices to show that $\mathsf{rec}_{\mathbf{1}}(f(\star))(z) = f(\mathsf{rec}_{\mathbf{1}}(\star)(z))$. Unpacking the definition of the recursor, we have to show:

$$\mathsf{pr}_1(z)_C(f(\star)) = f(\mathsf{pr}_1(z)_{\mathbf{1}}(\star)).$$

Using Σ-induction, we let

$$z \equiv (\alpha : \prod_{X:\mathsf{Set}} X \to X, p : \prod_{(X,Y:\mathsf{Set})} \prod_{(f:X\to Y)} \alpha_Y \circ f = f \circ \alpha_X)$$

and show:

$$\alpha_C(f(\star)) = f(\alpha_{\mathbf{1}}(\star)).$$

But $\mathsf{happly}(p\mathbf{1}Cf, \star)$ inhabits this type.

□

As discussed in §2.3, we may use the results of this section to show that our impredicative encoding of the unit type satisfies the expected induction principle (in fact, we only need an η-rule that says any $f : 1 \to 1$ such that $f(\star) = \star$ is propositionally equal to $\mathsf{rec}_{\mathbf{1}}(\star)$). Using $\mathbf{1}$-induction, we can prove that any $z : 1$ is

propositionally equal to $\star$ (cf. [Uni13, §1.5]), though we can also prove it directly from Theorem 2.4.6:

$$
\begin{aligned}
z &= \mathsf{id}(z) \\
&= \mathsf{rec}_{\mathbf{1}}(\mathsf{id}(\star))(z) && \text{(by Theorem 2.4.6 and function extensionality)} \\
&= \mathsf{rec}_{\mathbf{1}}(\star)(z) \\
&= \mathsf{rec}_{\mathbf{1}}((\lambda(y:\mathbf{1}).\star)(\star))(z) \\
&= (\lambda(y:\mathbf{1}).\star)(z) && \text{(again by Theorem 2.4.6 and function extensionality)} \\
&= \star
\end{aligned}
$$

Again, we have given an impredicative encoding (as a subtype of the system F-like encoding) that satisfies the desired universal property.

Chapter 3

Initial Algebras for Endofunctors

3.1 Generalizing

As with system F, we are able to encode not only additional type-formers, but data types also. The data type that we shall concentrate on in this chapter is the type of natural numbers. That being said, the method that we shall use to give an encoding of the natural numbers is very general and can be used to give impredicative encodings of a wide array of inductive types. Note that we are forced to do something different with the natural numbers, at least, for the "naturality" trick from Chapter 2 cannot be used to sharpen the relevant system F-like encoding (the translation of Eq. (1.1.5)):

$$\prod_{X:\mathsf{Set}} (X \to X) \to X \to X$$

The problem here is that X occurs contravariantly as well as covariantly (which might lead one to consider "dinatural transformations" [DS70], though we pursue a different avenue).

We consider categories of algebras for endofunctors:

$$T : \mathcal{S}et \to \mathcal{S}et,$$

where, recall, $\mathcal{S}et$ is the category whose objects are those of type:

$$\mathsf{Set} :\equiv \sum_{X:\mathcal{U}} \mathsf{isSet}(X)$$

and whose morphisms are given by

$$\hom_{\mathcal{S}et}(X, Y) :\equiv X \to Y.$$

In category theory, given an arbitrary endofunctor T on a category $\mathbb{C}$,the category of algebras **TAlg** for T has as objects pairs $(C \in \mathbb{C}, \alpha : TC \to C)$, with a morphism $f : (C, \alpha) \to (D, \beta)$ a morphism $f : C \to D$ in $\mathbb{C}$ such that $f \circ \alpha = \beta \circ Tf$, that is, such that the following square commutes.

$$\begin{array}{ccc} TC & \xrightarrow{Tf} & TD \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\beta} \\ C & \xrightarrow[f]{} & D \end{array}$$

For algebras for endofunctors $T : \mathbf{Set} \to \mathbf{Set}$, there is the obvious forgetful functor:

$$U : \mathbf{TAlg} \to \mathbf{Set},$$

which forgets the algebra structure and returns the underlying set, ie. $U(C, \alpha) := C$ (and similarly for morphisms). We will build impredicative encodings of inductive types as initial objects in the category of algebras for endofunctors.

Let 0 be the initial T-algebra we seek and

$$y : \mathbf{TAlg}^{\mathrm{op}} \to \mathbf{Set}^{\mathbf{TAlg}}$$

be the covariant Yoneda embedding. Inspired by the remark at the end of §2.1, by the (covariant) Yoneda lemma, we have that

$$U0 \cong \mathrm{Hom}_{\mathbf{Set}^{\mathbf{TAlg}}}(y0, U).$$

If 0 is initial, then $y0 = \mathbf{TAlg}(0, -)$ is the constant functor sending any T-algebra to the singleton set, which is the terminal object 1 in $\mathbf{Set}^{\mathbf{TAlg}}$. Thus:

$$U0 \cong \mathrm{Hom}_{\mathbf{Set}^{\mathbf{TAlg}}}(1, U)$$

But $1 = \Delta 1$ (the 1 on the right-hand side of the equation being the terminal object in **Set**), where $\Delta : \mathbf{Set} \to \mathbf{Set}^{\mathbf{TAlg}}$ is the functor defined by

$$\Delta X(\phi) := X.$$

So

$$U0 \cong \mathrm{Hom}_{\mathbf{Set}^{\mathbf{TAlg}}}(\Delta 1, U).$$

But Δ is left adjoint to the limit functor:

$$\varprojlim_{\phi \in \mathbf{TAlg}} (-)\phi : \mathbf{Set}^{\mathbf{TAlg}} \to \mathbf{Set}$$

Therefore, we have:

$$\begin{aligned} U0 &\cong \mathrm{Hom}_{\mathbf{Set}}\left(1, \varprojlim_{\phi \in \mathbf{TAlg}} U\phi\right) \\ &\cong \varprojlim_{\phi \in \mathbf{TAlg}} U\phi \end{aligned}$$

Using this fact, we have a *proposal* for a *definition* of the underlying set of the initial algebra 0:

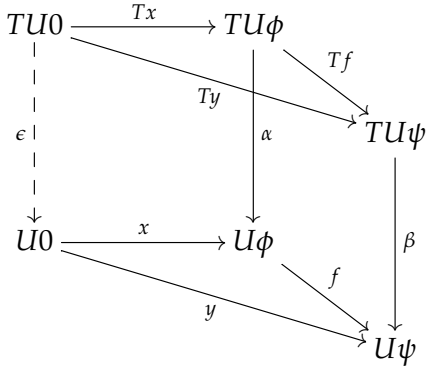
$$U0 := \varprojlim_{\phi \in \mathbf{TAlg}} U\phi \tag{3.1.1}$$

Given the knowledge that U is a right adjoint (the left adjoint being the free functor), we might have arrived at the same proposal by representing the initial object in **TAlg** as

$$0 = \varprojlim_{\phi \in \mathbf{TAlg}} \phi,$$

then utilizing the fact that right adjoints preserve limits.

Next we need to equip $U0$ from Eq. (3.1.1) with a T-algebra structure. As $U0$ is defined to be the limit of the forgetful functor, we have a map $x : U0 \to U\phi$ for any $\phi \in \mathbf{TAlg}$. Moreover, for any T-algebra homomorphism $f : (U\phi, \alpha) \to (U\psi, \beta)$ we have that the bottom triangle in the following diagram commutes.



(3.1.2)

As T is a functor, the top triangle also commutes and the square on the right commutes as f is an algebra homomorphism. Together this means that $TU0$ is a cone to the diagram U; but as $U0$ is the terminal such cone (as the limit), we get a unique map $\epsilon : TU0 \to U0$, which is the desired T-algebra structure on $U0$. We know that the forgetful functor U creates limits, which guarantees that the so-constructed T-algebra, $(U0, \epsilon)$, is an initial object.

3.2 The Limit in Type Theory

For the purposes of giving impredicative encodings in type theory, we want an explicit rendering of the limit $U0$ from Eq. (3.1.1). Given an endofunctor

$$T : \mathcal{S}et \to \mathcal{S}et,$$

we can write down the type of T-algebras and the type of T-algebra homomorphisms, which just express the usual categorical notions in type theory (cf. §3.1).

Definition 3.2.1.

Given an endofunctor $T : \mathcal{S}et \to \mathcal{S}et$, define the **type of T-algebras**:

$$\mathsf{TAlg} :\equiv \sum_{X:\mathsf{Set}} T(X) \to X,$$

and the **type of T-algebra homomorphisms** between T-algebras ϕ and ψ:

$$\mathsf{THom}(\phi, \psi) :\equiv \sum_{f:\mathsf{pr}_1(\phi)\to\mathsf{pr}_1(\psi)} \mathsf{pr}_2(\psi) \circ T(f) = f \circ \mathsf{pr}_2(\phi).$$

◇

Together these form a category $\mathcal{TA}lg$. We can also define the forgetful functor.

Definition 3.2.2.

Define the **forgetful functor** $U : \mathcal{TA}lg \to \mathcal{S}et$:

$$\begin{aligned} U(\phi) &:\equiv \mathsf{pr}_1(\phi) \\ U(\mu : \mathsf{THom}(\phi, \psi)) &:\equiv \mathsf{pr}_1(\mu) \end{aligned}$$

◇

We build the limit of the diagram U using products and equalizers in the usual way (cf. Proposition 5.21 in [Awo10]). The "product" (in type theory, dependent function)

$$\prod_{\phi:\mathsf{TAlg}} U(\phi) \tag{3.2.3}$$

has the right kind of "projections" $\prod_{(\phi:\mathsf{TAlg})} U(\phi) \to U(\psi)$ to be the limit of U, but these projections will not, in general, commute with arrows $U(\mu) : U(\phi) \to U(\psi)$ of the diagram. Thus we also consider the product over all T-algebra homomorphisms:

$$\prod_{(\phi,\psi:\mathsf{TAlg})} \prod_{(\mu:\mathsf{THom}(\phi,\psi))} U(\psi) \tag{3.2.4}$$

and two canonical maps from the the product in Eq. (3.2.3) to that in Eq. (3.2.4):

$$P_1 :\equiv \lambda\left(\Phi : \prod_{\phi:\mathsf{TAlg}} U(\phi)\right).\lambda(\phi:\mathsf{TAlg}).\lambda(\psi:\mathsf{TAlg}).\lambda(\mu:\mathsf{THom}(\phi,\psi)).\Phi(\psi)$$
$$P_2 :\equiv \lambda\left(\Phi : \prod_{\phi:\mathsf{TAlg}} U(\phi)\right).\lambda(\phi:\mathsf{TAlg}).\lambda(\psi:\mathsf{TAlg}).\lambda(\mu:\mathsf{THom}(\phi,\psi)).\mathsf{pr}_1(\mu)(\Phi(\phi))\,. \tag{3.2.5}$$

P_1 acts like a family of projections, whereas P_2 acts like a family of projections followed by (underlying functions of) T-algebra homomorphisms.

Taking the equalizer of P_1 and P_2 yields the subtype of the product in Eq. (3.2.3) such that the projections commute with arrows in the diagram.

$$E \overset{e}{\rightarrowtail} \prod_{(\phi:\mathsf{TAlg})} U(\phi) \underset{P_2}{\overset{P_1}{\rightrightarrows}} \prod_{(\phi,\psi:\mathsf{TAlg})} \prod_{(\mu:\mathsf{THom}(\phi,\psi))} U(\psi)$$

$$E :\equiv \sum_{\Phi:\prod_{(\phi:\mathsf{TAlg})} U(\phi)} P_1(\Phi) = P_2(\Phi) \tag{3.2.6}$$

Of course, the equalizing map e is just the first projection. Note that we could not do this in system F for there are no identity types there.

Observe that the identity type $P_1(\Phi) = P_2(\Phi)$ is a proposition and the equalizer E is a set.

The preceding construction indicates that *any* endofunctor on $\mathcal{S}et$—whose type of objects, recall, is a subtype of $\mathcal{U}$—has an initial algebra. The initial algebra is given by (E, ϵ), where E is the equalizer from Eq. (3.2.6) and ϵ is the algebra structure as determined in Eq. (3.1.2). Essential to this is the impredicativity of $\mathcal{U}$. Specifically, this guarantees that the product over TAlg lands in $\mathcal{U}$, given that TAlg is a subtype of Set and thus of $\mathcal{U}$. One should therefore be able to show that *any* endofunctor on $\mathcal{S}et$ has an initial algebra internal to the type theory. It is interesting to compare this to [Hyl88, §3.1], where is it shown that for any small complete category $\mathbb{C}$, there is an initial T-algebra for any endofunctor $T : \mathbb{C} \to \mathbb{C}$. Furthermore, we deduce that in this setting, we cannot define a powerset functor on $\mathcal{S}et$, for otherwise Girard's paradox arises (cf. [SU06]).

Finally, observe that in this setting, Lemma 2.1.5 arises as the special case for the endofunctor $T(X) :\equiv A$. Notably, the product in Eq. (3.2.3) can be written:

$$\begin{aligned}\prod_{\phi:\mathsf{TAlg}} U(\phi) &\equiv \prod_{\phi:\sum_{(X:\mathsf{Set})} T(X)\to X} U(\phi)\\ &\equiv \prod_{(X:\mathsf{Set})} \prod_{(\alpha:T(X)\to X)} X\\ &\equiv \prod_{X:\mathsf{Set}} (T(X)\to X)\to X\\ &\equiv \prod_{X:\mathsf{Set}} (A\to X)\to X.\end{aligned}$$

3.3 The Natural Numbers

As we are interested in obtaining an impredicative encoding of the natural numbers, the following endofunctor on $\mathcal{S}et$ is the relevant one.

Definition 3.3.1.

Define the endofunctor $N : \mathcal{S}et \to \mathcal{S}et$:

$$N(X) :\equiv X + \mathbf{1}$$
$$N(f : X \to Y) :\equiv \mathsf{rec}_+(\mathsf{inl} \circ f, \mathsf{inr}) : X + \mathbf{1} \to Y + \mathbf{1}$$

◇

With this, we have a type of N-algebras and a type of N-algebra homomorphisms, following Definition 3.2.1, which form a category $\mathcal{N}\mathcal{A}lg$; as well as the forgetful functor $\mathcal{N}\mathcal{A}lg \to \mathcal{S}et$, following Definition 3.2.2.

A nice feature of this development is that we may treat any coproduct (of sets) and $\mathbf{1}$ to be the impredicative encodings from Chapter 2. This means that, when we are done, we will have built the type of natural numbers impredicatively just from Σ-types, identity types and, of course, large Π-types over an impredicative universe $\mathcal{U}$.

Definition 3.3.2.

Following Eq. (3.2.6), we define the **type of natural numbers** as follows:

$$\mathbb{N} :\equiv \sum_{\Phi : \prod_{(\phi : \mathsf{NAlg})} U(\phi)} P_1(\Phi) = P_2(\Phi)\,,$$

where P_1 and P_2, from Eq. (3.2.5), are adapted to the case of the endofunctor N. ◇

Again, this encoding is (equivalent to) a subtype of the system F-like encoding. That is:

$$\prod_{\phi : \mathsf{NAlg}} U(\phi) \simeq \prod_{X : \mathsf{Set}} (X \to X) \to X \to X,$$

which is seen by a similar argument to that at the end of §3.2. As discussed at the end of §3.2, $\mathbb{N}$ is a set; this is crucial for what follows.

Next we endow $\mathbb{N}$ with a N-algebra structure. We define the constructors for $\mathbb{N}$: a zero element $0 : \mathbb{N}$ and a successor function $\mathsf{succ} : \mathbb{N} \to \mathbb{N}$. This is tantamount to showing that the impredicative encoding of $\mathbb{N}$ in Definition 3.3.2 satisfies the expected introduction rules. The full set of rules for $\mathbb{N}$ are in Figure 3.1. The algebra structure is then given by $\mathsf{rec}_+(\mathsf{succ}, \mathsf{rec}_{\mathbf{1}}(0)) : \mathbb{N} + \mathbf{1} \to \mathbb{N}$.

Moreover, we show that

$$(\mathbb{N} : \mathsf{Set}, \mathsf{rec}_+(\mathsf{succ}, \mathsf{rec}_{\mathbf{1}}(0)) : \mathbb{N} + \mathbf{1} \to \mathbb{N})$$

$$\frac{}{\Gamma \vdash \mathbb{N} : \mathsf{Set}}\ \mathbb{N}\text{-FORM}$$

$$\frac{}{\Gamma \vdash 0 : \mathbb{N}}\ \mathbb{N}\text{-INTRO}_1 \qquad \frac{}{\Gamma \vdash \mathsf{succ} : \mathbb{N} \to \mathbb{N}}\ \mathbb{N}\text{-INTRO}_2$$

$$\frac{\Gamma \vdash C : \mathsf{Set} \qquad \Gamma \vdash e : C \to C \qquad \Gamma \vdash c : C}{\Gamma \vdash \mathsf{rec}_{\mathbb{N}}(e, c) : \mathbb{N} \to C}\ \mathbb{N}\text{-ELIM}$$

$$\frac{\Gamma \vdash C : \mathsf{Set} \qquad \Gamma \vdash e : C \to C \qquad \Gamma \vdash c : C}{\Gamma \vdash \mathsf{rec}_{\mathbb{N}}(e, c)(0) \equiv c : C}\ \mathbb{N}\text{-}\beta_1$$

$$\frac{\Gamma \vdash C : \mathsf{Set} \qquad \Gamma \vdash e : C \to C \qquad \Gamma \vdash c : C \qquad \Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathsf{rec}_{\mathbb{N}}(e, c)(\mathsf{succ}(n)) \equiv e(\mathsf{rec}_{\mathbb{N}}(e, c)(n)) : C}\ \mathbb{N}\text{-}\beta_2$$

$$\frac{\begin{array}{c}\Gamma \vdash C : \mathsf{Set} \qquad \Gamma \vdash e : C \to C \qquad \Gamma \vdash c : C \qquad \Gamma \vdash f : \mathbb{N} \to C \\ \Gamma \vdash p_1 : f(0) = c \qquad \Gamma, x : \mathbb{N} \vdash p_2 : f(\mathsf{succ}(x)) = e(f(x))\end{array}}{\Gamma \vdash p_3 : \mathsf{rec}_{\mathbb{N}}(e, c) = f}\ \mathbb{N}\text{-}\eta$$

Figure 3.1: Inference rules for $\mathbb{N}$.

is a *weakly* initial algebra in $\mathcal{NA}lg$. That is, given set C, an endomap, $e : C \to C$ and a point $c : C$, we define the recursor $\mathsf{rec}_{\mathbb{N}}(e, c) : \mathbb{N} \to C$, which is tantamount to showing that $\mathbb{N}$ satisfies the expected simple elimination rule *with respect to other sets*. Later (Corollary 3.3.11) we will see that $(\mathbb{N}, \mathsf{rec}_+(\mathsf{succ}, \mathsf{rec}_{\mathbf{1}}(0)))$ is indeed an *initial* algebra (in the sense that any N-algebra homomorphism $\mathbb{N} \to C$ is propositionally equal to $\mathsf{rec}_{\mathbb{N}}(e, c)$). This follows from the fact that the β and η-computation rules are satisfied (Lemma 3.3.6 and Theorem 3.3.9, respectively).

Lemma 3.3.3.

We can define:

(i) a zero element $0 : \mathbb{N}$,
(ii) a successor function $\mathsf{succ} : \mathbb{N} \to \mathbb{N}$,
(iii) a recursor $\mathsf{rec}_{\mathbb{N}}(e, c) : \mathbb{N} \to C$, *for any* $C : \mathsf{Set}$, $e : C \to C$ *and* $c : C$.

Proof. (i)

$$\begin{aligned} 0 :\equiv\ & (\lambda(\phi : \mathsf{NAlg}).\mathsf{pr}_2(\phi)\mathsf{inr}(\star), \\ & \lambda(\phi : \mathsf{NAlg}).\lambda(\psi : \mathsf{NAlg}).\lambda(\mu : \mathsf{NHom}(\phi, \psi)).\mathsf{refl}_{\mathsf{pr}_2(\psi)\mathsf{inr}(\star)}) : \mathbb{N} \end{aligned}$$

(ii)

$$\mathsf{succ}(n) :\equiv (\lambda(\phi : \mathsf{NAlg}).\mathsf{pr}_2(\phi)\mathsf{inl}(\mathsf{pr}_1(n)(\phi)), p_{\mathsf{succ}}) : \mathbb{N}\,,$$

where the path

$$p_{\mathsf{succ}} : P_1(\lambda\phi.\mathsf{pr}_2(\phi)\mathsf{inl}(\mathsf{pr}_1(n)(\phi))) = P_2(\lambda\phi.\mathsf{pr}_2(\phi)\mathsf{inl}(\mathsf{pr}_1(n)(\phi)))$$

is obtained by the following argument.
Unpacking the definitions of P_1 and P_2, the desired path p_{succ} is of type

$$\lambda\phi.\lambda\psi.\lambda\mu.\mathsf{pr}_2(\psi)\mathsf{inl}(\mathsf{pr}_1(n)(\psi)) = \lambda\phi.\lambda\psi.\lambda\mu.\mathsf{pr}_1(\mu)(\mathsf{pr}_2(\phi)\mathsf{inl}(\mathsf{pr}_1(n)(\phi))).$$

But we have a path:

$$\mathsf{pr}_2(n) : \lambda\phi.\lambda\psi.\lambda\mu.\mathsf{pr}_1(n)(\psi) = \lambda\phi.\lambda\psi.\lambda\mu.\mathsf{pr}_1(\mu)(\mathsf{pr}_1(n)(\phi))$$

and thus a path:

$$q :\equiv \mathsf{happly}(\mathsf{happly}(\mathsf{happly}(\mathsf{pr}_2(n),\phi),\psi),\mu) : \mathsf{pr}_1(n)(\psi) = \mathsf{pr}_1(\mu)(\mathsf{pr}_1(n)(\phi)).$$

We can therefore obtain a path:

$$\mathsf{ap}_{\mathsf{pr}_2(\psi)\circ\mathsf{inl}}(q) : \mathsf{pr}_2(\psi)\mathsf{inl}(\mathsf{pr}_1(n)(\psi)) = \mathsf{pr}_2(\psi)(\mathsf{inl}(\mathsf{pr}_1(\mu)(\mathsf{pr}_1(n)(\phi)))). \tag{3.3.4}$$

Looking at the right-hand side of the above equality, we know that:

$$\begin{aligned}&\mathsf{pr}_2(\psi)(\mathsf{inl}(\mathsf{pr}_1(\mu)(\mathsf{pr}_1(n)(\phi))))\\ &\equiv \mathsf{pr}_2(\psi)(\mathsf{rec}_+(\mathsf{inl}\circ\mathsf{pr}_1(\mu),\mathsf{inr})(\mathsf{inl}(\mathsf{pr}_1(n)(\phi))))\\ &\equiv \mathsf{pr}_2(\psi)(N(\mathsf{pr}_1(\mu))(\mathsf{inl}(\mathsf{pr}_1(n)(\phi)))).\end{aligned}$$

As μ is an N-algebra homomorphism, $\mathsf{pr}_2(\mu)$ gives us a path of type $\mathsf{pr}_2(\psi)\circ N(\mathsf{pr}_1(\mu)) = \mathsf{pr}_1(\mu)\circ\mathsf{pr}_2(\phi)$. From this we can get a path:

$$\begin{aligned}r :\equiv &\mathsf{happly}(\mathsf{pr}_2(\mu),\mathsf{inl}(\mathsf{pr}_1(n)(\phi)))\\ &: \mathsf{pr}_2(\psi)(N(\mathsf{pr}_1(\mu))(\mathsf{inl}(\mathsf{pr}_1(n)(\phi)))) = \mathsf{pr}_1(\mu)(\mathsf{pr}_2(\phi)(\mathsf{inl}(\mathsf{pr}_1(n)(\phi))))).\end{aligned} \tag{3.3.5}$$

Composing the paths from Eq. (3.3.4) and Eq. (3.3.5) gives:

$$\mathsf{ap}_{\mathsf{pr}_2(\psi)\circ\mathsf{inl}}(q)\cdot r : \mathsf{pr}_2(\psi)\mathsf{inl}(\mathsf{pr}_1(n)(\psi)) = \mathsf{pr}_1(\mu)(\mathsf{pr}_2(\phi)(\mathsf{inl}(\mathsf{pr}_1(n)(\phi)))).$$

So, indeed:

$$\lambda\phi.\lambda\psi.\lambda\mu.\mathsf{ap}_{\mathsf{pr}_2(\psi)\circ\mathsf{inl}}(q)\cdot r : \prod_{(\phi,\psi:\mathsf{NAlg})}\prod_{(\mu:\mathsf{NHom}(\phi,\psi))} P_1\Phi\phi\psi\mu = P_2\Phi\phi\psi\mu$$

and thus we define:

$$p_{\mathsf{succ}} :\equiv \mathsf{funext}(\lambda\phi.\lambda\psi.\lambda\mu.\mathsf{ap}_{\mathsf{pr}_2(\psi)\circ\mathsf{inl}}(q)\cdot r).$$

(iii) Given any $C : \mathsf{Set}$, endomap $e : C \to C$ and point $c : C$, define:

$$\mathsf{rec}_{\mathbb{N}}(e, c) :\equiv \lambda(n : \mathbb{N}).\mathsf{pr}_1(n)(C, \mathsf{rec}_+(e, \mathsf{rec}_{\mathbf{1}}(c))) : \mathbb{N} \to C.$$

□

The constructors and the eliminator satisfy the expected β-rules for $\mathbb{N}$.

Lemma 3.3.6 - β-rules for $\mathbb{N}$.

Let $0 : \mathbb{N}$, $\mathsf{succ} : \mathbb{N} \to \mathbb{N}$ *and the recursor be as in Lemma 3.3.3,* $C : \mathsf{Set}$, $e : C \to C$, $c : C$ *and* $n : \mathbb{N}$. *Then the following hold definitionally.*

(i) $\mathsf{rec}_{\mathbb{N}}(e, c)(0) \equiv c$
(ii) $\mathsf{rec}_{\mathbb{N}}(e, c)(\mathsf{succ}(n)) \equiv e(\mathsf{rec}_{\mathbb{N}}(e, c)(n))$

Proof. (i)

$$\begin{aligned}
&\mathsf{rec}_{\mathbb{N}}(c, f)(0) \\
&\equiv \lambda n.\mathsf{pr}_1(n)(C, \mathsf{rec}_+(e, \mathsf{rec}_{\mathbf{1}}(c)))(\lambda\phi.\mathsf{pr}_2(\phi)(\mathsf{inr}(\star)), \lambda\phi.\lambda\psi.\lambda\mu.\mathsf{refl}_{\mathsf{pr}_2(\psi)\mathsf{inr}(\star)}) \\
&\equiv \mathsf{pr}_1(\lambda\phi.\mathsf{pr}_2(\phi)(\mathsf{inr}(\star)), \lambda\phi.\lambda\psi.\lambda\mu.\mathsf{refl}_{\mathsf{pr}_2(\psi)\mathsf{inr}(\star)})(C, \mathsf{rec}_+(e, \mathsf{rec}_{\mathbf{1}}(c))) \\
&\equiv \lambda\phi.\mathsf{pr}_2(\phi)(\mathsf{inr}(\star))(C, \mathsf{rec}_+(e, \mathsf{rec}_{\mathbf{1}}(c))) \\
&\equiv \mathsf{pr}_2(C, \mathsf{rec}_+(e, \mathsf{rec}_{\mathbf{1}}(c)))(\mathsf{inr}(\star)) \\
&\equiv \mathsf{rec}_+(e, \mathsf{rec}_{\mathbf{1}}(c))(\mathsf{inr}(\star)) \\
&\equiv \mathsf{rec}_{\mathbf{1}}(c)(\star) && \text{(by Lemma 2.2.5, the } \beta\text{-rule for the coproduct)} \\
&\equiv c && \text{(by Lemma 2.4.3, the } \beta\text{-rule for } \mathbf{1})
\end{aligned}$$

(ii)

$$\begin{aligned}
&\mathsf{rec}_{\mathbb{N}}(e, c)(\mathsf{succ}(n)) \\
&\equiv \lambda n.\mathsf{pr}_1(n)(C, \mathsf{rec}_+(e, \mathsf{rec}_{\mathbf{1}}(c)))(\lambda\phi.\mathsf{pr}_2(\phi)(\mathsf{inl}(\mathsf{pr}_1(n)(\phi)), p_{\mathsf{succ}})) \\
&\equiv \mathsf{pr}_1(\lambda\phi.\mathsf{pr}_2(\phi)(\mathsf{inl}(\mathsf{pr}_1(n)(\phi))), p_{\mathsf{succ}})(C, \mathsf{rec}_+(e, \mathsf{rec}_{\mathbf{1}}(c))) \\
&\equiv \lambda\phi.\mathsf{pr}_2(\phi)(\mathsf{inl}(\mathsf{pr}_1(n)(\phi)))(C, \mathsf{rec}_+(e, \mathsf{rec}_{\mathbf{1}}(c))) \\
&\equiv \mathsf{rec}_+(e, \mathsf{rec}_{\mathbf{1}}(c))(\mathsf{inl}(\mathsf{pr}_1(n)(C, \mathsf{rec}_+(e, \mathsf{rec}_{\mathbf{1}}(c))))) \\
&\equiv e(\mathsf{pr}_1(n)(C, \mathsf{rec}_+(e, \mathsf{rec}_{\mathbf{1}}(c)))) \\
&\qquad\qquad \text{(by Lemma 2.2.5, the } \beta\text{-rule for the coproduct)} \\
&\equiv e(\mathsf{rec}_{\mathbb{N}}(e, c)(n))
\end{aligned}$$

□

We now prove a key lemma towards showing that $(\mathbb{N}, \mathsf{rec}_+(\mathsf{succ}, \mathsf{rec}_{\mathbf{1}}(0)))$ is an initial N-algebra. The lemma is for $\mathbb{N}$ what Lemma 2.2.6 is for the coproduct and Lemma 2.4.4 is for $\mathbf{1}$, a weak η-rule.

Lemma 3.3.7 - weak η-rule for $\mathbb{N}$.

The following propositional equality holds for any $n : \mathbb{N}$.

$$\mathsf{rec}_{\mathbb{N}}(0, \mathsf{succ})(n) = n$$

Proof. Unpacking $\mathsf{rec}_{\mathbb{N}}(0, \mathsf{succ})(n)$, we have to show

$$\mathsf{pr}_1(n)(\mathbb{N}, \mathsf{rec}_+(\mathsf{succ}, \mathsf{rec}_{\mathbf{1}}(0))) = n.$$

By Lemma 2.7.2 of [Uni13], it suffices to show that:

(i) we have a path $p : \mathsf{pr}_1(\mathsf{pr}_1(n)(\mathbb{N}, \mathsf{rec}_+(\mathsf{succ}, \mathsf{rec}_{\mathbf{1}}(0)))) = \mathsf{pr}_1(n)$,
(ii) $p_*(\mathsf{pr}_2(\mathsf{pr}_1(n)(\mathbb{N}, \mathsf{rec}_+(\mathsf{succ}, \mathsf{rec}_{\mathbf{1}}(0))))) = \mathsf{pr}_2(n)$.

We use Σ-induction to let $n \equiv (\Phi : \prod_{(\phi:\mathsf{NAlg})} U\phi, q : P_1(\Phi) = P_2(\Phi))$. Then, for (i), using function extensionality, we want to show:

$$(\mathsf{pr}_1(\Phi(\mathbb{N}, \mathsf{rec}_+(\mathsf{succ}, \mathsf{rec}_{\mathbf{1}}(0)))))(\phi) = \Phi(\phi)$$

for an arbitrary $\phi : \mathsf{NAlg}$.

Notice that $P_1(\Phi) \equiv \lambda\phi.\lambda\psi.\lambda\mu.\Phi(\psi)$ and $P_2(\Phi) \equiv \lambda\phi.\lambda\psi.\lambda\mu.\mathsf{pr}_1(\mu)(\Phi(\phi))$. We define a function:

$$f_\phi : U(\mathbb{N}, \mathsf{rec}_+(\mathsf{succ}, \mathsf{rec}_{\mathbf{1}}(0))) \equiv \mathbb{N} \to U(\phi)$$
$$f_\phi(n) :\equiv \mathsf{pr}_1(n)(\phi)$$

If f_ϕ is an N-algebra homomorphism, with r as the path witnessing this, then

$$(\mathsf{happly}(\mathsf{happly}(\mathsf{happly}(q, (\mathbb{N}, \mathsf{rec}_+(\mathsf{succ}, \mathsf{rec}_{\mathbf{1}}(0)))), \phi), (f_\phi, r)))^{-1}$$

is of type

$$f_\phi(\Phi(\mathbb{N}, \mathsf{rec}_+(\mathsf{succ}, \mathsf{rec}_{\mathbf{1}}(0)))) = \Phi(\phi),$$

ie. of type

$$\mathsf{pr}_1(\Phi(\mathbb{N}, \mathsf{rec}_+(\mathsf{succ}, \mathsf{rec}_1(0))))(\phi) = \Phi(\phi),$$

which is what we want to show.

So we now verify that f_ϕ is indeed an N-algebra homomorphism, ie. that

$$\mathsf{pr}_2(\phi) \circ N(f_\phi) = f_\phi \circ \mathsf{pr}_2(\mathbb{N}, \mathsf{rec}_+(\mathsf{succ}, \mathsf{rec}_{\mathbf{1}}(0))).$$

Using the definition of N and function extensionality, it suffices to show:

$$\mathsf{pr}_2(\phi)(\mathsf{rec}_+(\mathsf{inl} \circ f_\phi, \mathsf{inr})(x)) = f_\phi(\mathsf{rec}_+(\mathsf{succ}, \mathsf{rec}_{\mathbf{1}}(0))(x)) \tag{3.3.8}$$

for an arbitrary $x : \mathbb{N} + \mathbf{1}$.

By $+$-induction (Theorem 2.3.1), we can do a case analysis on x. First, let $x \equiv \mathsf{inl}(m : \mathbb{N})$. Then Eq. (3.3.8) becomes $\mathsf{pr}_2(\phi)(\mathsf{inl}(f_\phi(m))) = f_\phi(\mathsf{succ}(m))$. Looking at the right-hand side of this equation and using the definition of $\mathsf{succ}(m)$, we have:

$$\begin{aligned} f_\phi(\lambda\phi.\mathsf{pr}_2(\phi)(\mathsf{inl}(\mathsf{pr}_1(m)(\phi))), p_{\mathsf{succ}}) &\equiv (\lambda\phi.\mathsf{pr}_2(\phi)(\mathsf{inl}(\mathsf{pr}_1(m)(\phi))))(\phi) \\ &\equiv \mathsf{pr}_2(\phi)(\mathsf{inl}(\mathsf{pr}_1(m)(\phi))) \\ &\equiv \mathsf{pr}_2(\phi)(\mathsf{inl}(f_\phi(m))), \end{aligned}$$

as desired.

For the second case, also using $\mathbf{1}$-induction, let $x \equiv \mathsf{inr}(\star)$. Then Eq. (3.3.8) becomes $\mathsf{pr}_2(\phi)(\mathsf{inr}(\star)) = f_\phi(\mathsf{rec}_{\mathbf{1}}(0)(\star))$. But

$$f_\phi(\mathsf{rec}_{\mathbf{1}}(0)(\star)) \equiv f_\phi(0) \equiv \mathsf{pr}_1(0)(\phi) \equiv \mathsf{pr}_2(\phi)(\mathsf{inr}(\star)),$$

where the last step uses the definition of 0 from Lemma 3.3.3. Therefore, f_ϕ is an N-algebra homomorphism and we are done with (i).

(ii) simply follows from the fact that, being an equality over a set, $P_1(\Phi) = P_2(\Phi)$ is a proposition.

□

We now come to the main theorem—and corollary—of this section and, indeed, of this project: that our impredicative encoding of $\mathbb{N}$ satisfies the η-computation rule for $\mathbb{N}$—and so has the desired universal property. We call the theorem the η-rule as it expresses the *uniqueness clause* of the universal property of $\mathbb{N}$ and it arises as a special case of the η-rule for W-types as formulated in [AGS12] (reasons analogous to those in the case of the coproduct in §2.2).

Theorem 3.3.9 - η-rule for $\mathbb{N}$.

Let $C : \mathsf{Set}$, $e : C \to C$ and $c : C$. Moreover, let $f : \mathbb{N} \to C$ such that

$$\begin{aligned} f(0) &= c, \\ f(\mathsf{succ}(x)) &= e(f(x)), \end{aligned}$$

for any $x : \mathbb{N}$. Then

$$\mathsf{rec}_{\mathbb{N}}(e, c) = f.$$

Proof. First note that $f = f \circ \mathsf{rec}_{\mathbb{N}}(\mathsf{succ}, 0)$ by Lemma 3.3.7. So we show:

$$\mathsf{rec}_{\mathbb{N}}(e, c) = f \circ \mathsf{rec}_{\mathbb{N}}(\mathsf{succ}, 0).$$

Using the definition of the recursor, this amounts to showing:

$$\lambda n.\mathsf{pr}_1(n)(C, \mathsf{rec}_+(e, \mathsf{rec}_{\mathbf{1}}(c))) = f \circ (\lambda n.\mathsf{pr}_1(n)(\mathbb{N}, \mathsf{rec}_+(\mathsf{succ}, \mathsf{rec}_{\mathbf{1}}(0)))).$$

Next we use function extensionality and Σ-induction to give each side of the above equation an argument $n \equiv (\Phi : \prod_{(\phi:\mathsf{NAlg})} U\phi, s : P_1(\Phi) = P_2(\Phi))$. So now we have to show:

$$\Phi(C, \mathsf{rec}_+(e, \mathsf{rec}_{\mathbf{1}}(c))) = f(\Phi(\mathbb{N}, \mathsf{rec}_+(\mathsf{succ}, \mathsf{rec}_{\mathbf{1}}(0)))). \tag{3.3.10}$$

But the left-hand side of Eq. (3.3.10) is just

$$P_1\Phi(\mathbb{N}, \mathsf{rec}_+(\mathsf{succ}, \mathsf{rec}_{\mathbf{1}}(0)))(C, \mathsf{rec}_+(e, \mathsf{rec}_{\mathbf{1}}(c)))\mu$$

and the right-hand side is

$$P_2\Phi(\mathbb{N}, \mathsf{rec}_+(\mathsf{succ}, \mathsf{rec}_{\mathbf{1}}(0)))(C, \mathsf{rec}_+(e, \mathsf{rec}_{\mathbf{1}}(c)))\mu.$$

So

$$\mathsf{happly}(\mathsf{happly}(\mathsf{happly}(s, (\mathbb{N}, \mathsf{rec}_+(\mathsf{succ}, \mathsf{rec}_{\mathbf{1}}(0)))), (C, \mathsf{rec}_+(e, \mathsf{rec}_{\mathbf{1}}(c)))), \mu)$$

inhabits the type of Eq. (3.3.10).

□

The β-rules for $\mathbb{N}$ tell us that $\mathsf{rec}_{\mathbb{N}}(e, c)$ is an N-algebra homomorphism, for any $C : \mathsf{Set}$, $e : C \to C$ and $c : C$. Therefore, we may concisely sum up the results of this section in the following, which expresses that our impredicative encoding $\mathbb{N}$ has the universal property of the natural numbers.

Corollary 3.3.11 - universal property for $\mathbb{N}$.

Let $C : \mathsf{Set}$, $e : C \to C$ and $c : C$. Moreover, let μ be an N-algebra homomorphism, ie. let

$$\mu : \mathsf{NHom}((\mathbb{N}, \mathsf{rec}_+(\mathsf{succ}, \mathsf{rec}_{\mathbf{1}}(0))), (C, \mathsf{rec}_+(e, \mathsf{rec}_{\mathbf{1}}(c)))).$$

Then

$$(\mathsf{rec}_{\mathbb{N}}(e, c), p) = \mu,$$

where p exhibits $\mathsf{rec}_+(e, \mathsf{rec}_{\mathbf{1}}(c))$ as an N-algebra homomorphism. In other words, any N-algebra homomorphism is (propositionally) equal to the appropriate recursor (as an N-algebra homomorphism).

The paths p and $\mathsf{pr}_2(\mu)$ are equal as $\mathsf{rec}_+(e, \mathsf{rec}_{\mathbf{1}}(c)) \circ N(f) = f \circ \mathsf{rec}_+(\mathsf{succ}, \mathsf{rec}_{\mathbf{1}}(0))$ is a proposition.

The preceding results tell us that our impredicative encoding of $\mathbb{N}$ also satisfies the usual induction principle. This follows from the results in [AGS12], given that $\mathbb{N}$ can be formulated as a W-type, or can be proved directly as in §2.3.

3.4 Conclusion

Let us conclude this chapter by reflecting on the progress made. System F's Polynat:

$$\forall X.(X \to X) \to X \to X$$

satisfies the unusual formation, introduction, elimination and β-computation rules for $\mathbb{N}$, but does *not* always determine a natural numbers object. In [Rum04], Rummelhoff gives a PER model of system F in which it is not the case that the interpretation of Polynat, $\mathcal{N}$, contains only interpretations of 'church numerals', ie. terms of the form

$$\Lambda X.\lambda(f : X \to X).\lambda(x : X).f(f(...(f(x))))$$

(For more on PER models of system F, see [Cro94].) This means that, in general, a function defined by recursion on $\mathcal{N}$, if it exists, need not be unique. This makes it clear that Polynat violates the η-rule for $\mathbb{N}$.

On the other hand, what we have done here is constructed an impredicative encoding of the type of natural numbers satisfying the expected formation rule, introduction rule, simple elimination rule, β-computation rules (definitionally) and η-rule up to propositional equality—equivalently, the expected formation and introduction rules, the dependent elimination rule and the corresponding propositional β and η-rules. Semantically, this means that our impredicative encoding does *always* determine a natural numbers object.

Chapter 4

Future Work

To conclude, we briefly describe some avenues for further investigation in relation to this project:

- Of course, one could continue building up a "library" of impredicative encodings. An impredicative encoding of W-types is a natural next step, where it should be possible to use the methods of Chapter 3 with the following endofunctor on $\mathcal{S}et$.

  $$T(X) :\equiv \sum_{a:A} B(a) \to X$$

  Dually, one might look in to building *co*inductive types, eg. M-types, as *final co*algebras.

  There is also the possibility of building more exotic inductive types that may have a higher h-level. For example, one might want to build an impredicative encoding of the coproduct of 1-types A and B. In this case, a more elaborate naturality condition would be needed.

  One could consider higher inductive types, such as the circle $\mathbb{S}^1$, set-quotients and truncations.

  Finally, one might consider inductive families, eg. vectors of length n and even identity types.

  Note that for the types considered in this project, we have only used some of the power afforded by the impredicativity of the universe $\mathcal{U}$, that is, we only ever quantified over $\mathcal{U}$ itself. It is expected that building more exotic types will involve quantifying over arbitrary types, thereby utilizing the full power of the impredicative universe.
- As mentioned at the end of §3.2, it would be nice to formalize the argument that every endofunctor on $\mathcal{S}et$ has an initial algebra in the type theory.

- A final point which we believe warrants further investigation is in relation to the remark at the end of §2.3. In the terminology of Chapter 3: the equivalence between the simple and dependent elimination rules for an initial algebra ϕ of some endofunctor on $\mathcal{S}et$ apparently holds in the presence of a weaker η-rule which only posits uniqueness of endo-homomorphisms from ϕ to ϕ.

Appendix A

Semantics

In this appendix, we outline a realizability semantics for the system set out in §1.2 based on assemblies and modest sets. Our presentation very closely follows lecture notes of Streicher [Str08] (especially §6), which in turn are based on prior work of Hyland [Hyl88] and Carboni, Freyd and Scedrov [CFS88]. Specifically, we give semantics in the category of assemblies over an arbitrary partial combinatory algebra (pca). The category of assemblies over a pca is a categorical generalization of Kleene's original *number realizability* [Kle45], where the pca provides an abstract notion of computation. The same basic machinery can be used to give a model of system F, as one might expect.

In what follows, given a set A and a partial, binary operation $\cdot : A \times A \to A$, a polynomial over A is a formal term built up from variables ranging over A and constants—one for every element of A—using the operation $\cdot$. We abuse notation and use a, say, to stand for both the element of A and the constant denoting this element. For polynomials t and s over A, $t \downarrow$ means that $t[a/x]$ is defined (in terms of the operation $\cdot$) for every $a \in A$, and $t \simeq s$ means that either both t and s are undefined or else they are both defined and equal to each other.

Definition A.0.1.

Let $\mathcal{A} := (|\mathcal{A}|, \cdot)$, where $|\mathcal{A}|$ is a non-empty set and $\cdot : |\mathcal{A}| \times |\mathcal{A}| \rightharpoonup |\mathcal{A}|$ is a partial, binary operation. Then $\mathcal{A}$ is a **partial combinatory algebra** (pca) iff for every polynomial t over $|\mathcal{A}|$ and every variable x ranging over $|\mathcal{A}|$, there exists a polynomial $\lambda^* x.t$ over $|\mathcal{A}|$, with $\mathrm{FV}(\lambda^* x.t) \subseteq \mathrm{FV}(t) \setminus \{x\}$, such that $\lambda^* x.t \downarrow$ and $(\lambda^* x.t) \cdot a \simeq t[a/x]$ for all $a \in |\mathcal{A}|$. ◇

By convention, the operation $\cdot$ associates to the left and we often write ab for $a \cdot b$. One can also consult [vO08] for more details.

The canonical example of a pca is the first Kleene algebra, where the underlying set is the set of natural numbers and the partial operation is Kleene application.

We will make use of the fact that in any pca we can form the pair of any $a, b \in$

$|\mathcal{A}|$. That is, for any pca $\mathcal{A}$, there exist $\mathsf{p}, \mathsf{p}_1, \mathsf{p}_2 \in |\mathcal{A}|$ such that:

$$\begin{aligned} \mathsf{p}aa' &\downarrow, \\ \mathsf{p}_1(\mathsf{p}aa') &= a, \\ \mathsf{p}_2(\mathsf{p}aa') &= a', \end{aligned}$$

for all $a, a' \in |\mathcal{A}|$. The elements are given by:

$$\begin{aligned} \mathsf{p} &:= \lambda^* x.\lambda^* y.\lambda^* z.zxy, \\ \mathsf{p}_1 &:= \lambda^* z.z(\lambda^* x.\lambda^* y.x), \\ \mathsf{p}_2 &:= \lambda^* z.z(\lambda^* x.\lambda^* y.y). \end{aligned}$$

Note also that we have the "identity polynomial": $\mathsf{id} := \lambda^* x.x$.

Definition A.0.2.

Let $\mathcal{A}$ be any pca.

(i) We define the **category of assemblies Asm**$(\mathcal{A})$ over $\mathcal{A}$ as follows:

- As objects we take pairs $X = (|X|, \|\cdot\|_X)$, where $|X|$ is a set and $\|\cdot\| : |X| \to \mathcal{P}_{\geq 1}(|\mathcal{A}|)$ (where $\mathcal{P}_{\geq 1}$ denotes the powerset without the empty set). We write $a \Vdash_X x$ for $a \in \|x\|_X$, saying: "a realizes x," and often omit subscripts when the context makes it clear.
- A morphism $f : X \to Y$ is a set-theoretic function $f : |X| \to |Y|$ such that there exists $e \in |\mathcal{A}|$ such that for every $x \in |X|$ and every $a \Vdash x$ we have $e \cdot a \downarrow$ and $e \cdot a \Vdash_Y f(x)$—in which case we write $e \Vdash f$. Composition and identity morphisms are inherited from **Set**. Realizer-wise, identities are realized by $\lambda^* x.x$, and if $a \Vdash f : X \to Y$ and $a' \Vdash g : Y \to Z$, then $g \circ f$ is realized by $\lambda^* x.a'(ax)$.

(ii) We also define the full and faithful functor

$$\nabla : \mathbf{Set} \to \mathbf{Asm}(\mathcal{A}), \tag{A.0.3}$$

where $\nabla(S)$ is the assembly with $|\nabla(S)| := S$ and $\|s\|_{\nabla(S)} := |\mathcal{A}|$ for all $s \in S$, and $\nabla(f) := f$, which is realized by id.

(iii) An assembly X is a **modest set** iff $x = x'$ whenever $\|x\|_X \cap \|x'\|_X$ is non-empty. We write **Mod**$(\mathcal{A})$ for the full subcategory of **Asm**$(\mathcal{A})$ with objects the modest sets.

$\diamond$

For the rest of this section, we fix an arbitrary pca $\mathcal{A}$. The categories **Asm**$(\mathcal{A})$ and **Mod**$(\mathcal{A})$ have many nice properties. In particular, **Asm**$(\mathcal{A})$ has all finite limits

and is cartesian closed. Moreover, $\mathbf{Mod}(\mathcal{A})$ is closed under finite limits taken in $\mathbf{Asm}(\mathcal{A})$ and the exponential object $[X \to Y]$ of X and Y is modest whenever Y is.

We give some constructions (see [Fru16], also). The terminal object 1 is given by:

$$|1| := \{\star\},$$
$$\| \star \| := |\mathcal{A}|.$$

The unique map $!_X : X \to 1$ for any $X \in \mathbf{Asm}(\mathcal{A})$ is inherited from **Set**, with realizer id.

The product of assemblies X and Y, $X \times Y$, is given by:

$$|X \times Y| := |X| \times |Y|,$$
$$a \Vdash_{X\times Y} (x, y) \iff \mathsf{p}_1 a \Vdash_X x \text{ and } \mathsf{p}_2 a \Vdash_Y y.$$

The usual set-theoretic projections, $\pi_1 : |X \times Y| \to |X|$ and $\pi_2 : |X \times Y| \to |Y|$, are realized by p_0 and p_1.

The equalizer $e : E \to X$ of arrows $f, g : X \to Y$ is given by

$$|E| := \{x \in X \mid f(x) = g(x)\},$$

where e is the canonical inclusion, realized by id.

Pullbacks can be built using products and equalizers.

$$\begin{array}{ccc} X \times_Z Y & \xrightarrow{\pi_2} & Y \\ {\scriptstyle \pi_1}\downarrow & \lrcorner & \downarrow{\scriptstyle g} \\ X & \xrightarrow[f]{} & Z \end{array}$$

That is, $|X \times_Y Z| := \{(x, y) \in |X \times Y| \mid f(a) = g(c)\}$. The realizer agrees with that for products.

Finally, we construct exponentials in $\mathbf{Asm}(\mathcal{A})$. Let $X, Y \in \mathbf{Asm}(\mathcal{A})$. The exponential object $[X \to Y]$ is given by:

$$|[X \to Y]| := \mathrm{Hom}_{\mathbf{Asm}(\mathcal{A})}(X, Y),$$
$$\|f : X \to Y\|_{[X\to Y]} := \{e \in |\mathcal{A}| \mid e \Vdash f\}.$$

The evaluation map $\epsilon : [X \to Y] \times X \to Y$, which sends $(f, x) \mapsto f(x)$, is realized by $\lambda^* x.\mathsf{p}_1 x(\mathsf{p}_2 x)$. If Y is modest, then if $e \Vdash f$ and $e \Vdash g$, then $f = g$—thus, $[X \to Y]$ is modest.

$\mathbf{Asm}(\mathcal{A})$ and $\mathbf{Mod}(\mathcal{A})$ give rise to models of first-order logic. Specifically, for any morphism $f : Y \to X$ in $\mathbf{Asm}(\mathcal{A})$, we have left and right adjoints to the monotone map $f^* : \mathrm{Sub}(X) \to \mathrm{Sub}(Y)$, given by pullback, which each satisfy the relevant Beck-Chevalley condition. This restricts to $\mathbf{Mod}(\mathcal{A})$. However, in both cases, the result can be strengthened, as follows.

Theorem A.0.4.

For every morphism $f : X \to Y$ in $\mathbf{Asm}(\mathcal{A})$, the pullback functor

$$f^* : \mathbf{Asm}(\mathcal{A})/Y \to \mathbf{Asm}(\mathcal{A})/X$$

has left and right adjoints:

$$\Sigma_f \dashv f^* \dashv \Pi_f.$$

Moreover, Σ_f and Π_f each satisfy the relevant Beck-Chevalley condition. That is, for any pullback square

$$\begin{array}{ccc} U & \xrightarrow{g} & X \\ {\scriptstyle p}\downarrow & \lrcorner & \downarrow{\scriptstyle f} \\ V & \xrightarrow[q]{} & Y \end{array}$$

and $m \in \mathbf{Asm}(\mathcal{A})/X$, we have:

$$\Sigma_p g^* m \cong q^* \Sigma_f m,$$
$$q^* \Pi_f m \cong \Pi_p g^* m.$$

All of the above restricts to $\mathbf{Mod}(\mathcal{A})$ also.

Proof. We give the constructions of Σ_f and Π_f, starting with the former. Given an object $g : U \to X$ of $\mathbf{Asm}(\mathcal{A})/X$, set:

$$\Sigma_f g := f \circ g.$$

For a morphism $k : g' \to g$ in $\mathbf{Asm}(\mathcal{A})/X$, set:

$$\Sigma_f(k : g' \to g) := k : f \circ g' \to f \circ g$$

in $\mathbf{Asm}(\mathcal{A})/Y$.

Now for the right adjoint. Let $g : U \to X$ be an object in $\mathbf{Asm}(\mathcal{A})/X$; we construct

$$\Pi_f g : P \to Y$$

as follows. Let P_0 be the set of all pairs (y, s) such that $y \in |Y|$ and $s : f^{-1}(y) \to |U|$ with $g(s(x)) = x$ for all $x \in f^{-1}(y)$. Set: $e \Vdash (y, s)$ iff $\mathsf{p}_1 e \Vdash_Y y$, $\mathsf{p}_2 e \downarrow$ and $\mathsf{p}_2 ea \Vdash_U s(x)$ whenever $a \Vdash_X x \in f^{-1}(y)$. Define P as the assembly with:

$$|P| := \{(y, s) \in P_0 \mid \exists e \in |\mathcal{A}|.\, e \Vdash (y, s)\},$$
$$\|(y, s)\|_P := \{e \in |\mathcal{A}| \mid e \Vdash (y, s)\}.$$

Finally, we define:

$$(\Pi_f g)(y, s) := y,$$

with realizer p_1.

If $g' : U' \to X$ is an object in $\mathbf{Asm}(\mathcal{A})$ and g is as above, the action of Π_f on a morphism $h : g \to g'$ is given by the following:

$$(\Pi_f h)(y, s) := (y, h \circ s),$$
$$\mathsf{p}(\mathsf{p}_1 e)(\lambda^* x.e'(\mathsf{p}_2 e x)) \Vdash (\Pi_f h)(y, s),$$

where $e \Vdash (y, s)$ and $e' \Vdash h$. □

A corollary of this is that $\mathbf{Asm}(\mathcal{A})$ is locally cartesian closed, which means that all its slices are cartesian closed. As a result, (up to the usual coherence issues) we can interpret Martin-Löf type theory with Σ-types; (extensional) identity types and (ordinary) Π-types in $\mathbf{Asm}(\mathcal{A})$, following Seely [See84]. What distinguishes the model in assemblies is that it admits an interpretation of the impredicative universe $\mathcal{U}$ using a special class of maps called "families of modest sets" (or "modest families").

Definition A.0.5.

A **family of modest sets** in $\mathbf{Asm}(\mathcal{A})$, indexed by an assembly X, is a morphism $\tilde{b} : B \to X$ in $\mathbf{Asm}(\mathcal{A})$ such that for all $x : 1 \to X$ the object B_x in the following pullback square is a modest set.

$$\begin{array}{ccc} B_x & \longrightarrow & B \\ {\scriptstyle x^*\tilde{b}}\downarrow & \lrcorner & \downarrow{\scriptstyle \tilde{b}} \\ 1 & \xrightarrow[x]{} & X \end{array} \tag{A.0.6}$$

We write $\mathbf{Mod}(\mathcal{A})(X)$ for the full subcategory of the slice category $\mathbf{Asm}(\mathcal{A})/X$ whose objects are families of modest sets indexed by X. ◇

The following is a useful characterization of families of modest sets.

Lemma A.0.7.

A morphism $\tilde{b} : B \to X$ in $\mathbf{Asm}(\mathcal{A})$ is a family of modest sets iff for any $b_1, b_2 \in B$, $b_1 = b_2$ whenever $\tilde{b}(b_1) = \tilde{b}(b_2)$ and $||b_1||_B \cap ||b_2||_B \neq \emptyset$.

Proof. Recall the construction of pullbacks above: elements of $|B_x|$ (as in the diagram in Eq. (A.0.6)) are pairs $(\star, b) \in 1 \times B$ such that $x(\star) = \tilde{b}(b)$.

First, let $\tilde{b}$ be a modest family. Now assume $\tilde{b}(b_1) = \tilde{b}(b_2)$ and $e \Vdash b_1, b_2$. We know $a \Vdash \star$ for any $a \in |\mathcal{A}|$. So $\mathsf{p}ae \Vdash (\star, b_1), (\star, b_2)$ for all $a \in |\mathcal{A}|$. As B_x is modest, we have $(\star, b_1) = (\star, b_2)$, which implies $b_1 = b_2$.

Conversely, assume $b_1 = b_2$ whenever $\tilde{b}(b_1) = \tilde{b}(b_2)$ and $||b_1||_B \cap ||b_2||_B \neq \emptyset$. We want to show that B_x is modest. On the one hand, assume $e \Vdash (\star, b_1), (\star, b_2)$. Then $\mathsf{p}_2 e \Vdash b_1, b_2$. Also, as $(\star, b_1), (\star, b_2) \in B_x$, we have $\tilde{b}(b_1) = x(\star) = \tilde{b_2}$. So,

by our assumption, $b_1 = b_2$ as required. On the other hand, let $(\star, b_1) = (\star, b_2)$. Then $b_1 = b_2$. So there is some $e \Vdash b_1, b_2$ and we have $\tilde{b}(b_1) = \tilde{b}(b_2)$. So $\mathsf{p}ae \Vdash (\star, b_1), (\star, b_2)$ for any $a \in |\mathcal{A}|$. □

Now for a crucial theorem.

Theorem A.0.8.

For every morphism $f : X \to Y$ in $\mathbf{Asm}(\mathcal{A})$, the functor Π_f preserves families of modest sets, that is, if $\tilde{b} : B \to X$ is a modest family, then so is $\Pi_f \tilde{b}$.

Proof. Assume $\tilde{b} : B \to X$ is a family of modest sets and recall the construction of Π_f from Theorem A.0.4: elements in the domain of $\Pi_f \tilde{b}$ are of the form $(y \in |Y|, s : f^{-1}(y) \to |B|)$ and $(\Pi_f \tilde{b})(y, s) = y$. By the right-to-left direction of Lemma A.0.7, we assume that we have $e \Vdash (y, s_1), (y, s_2)$ and show $s_1 = s_2$ (we can assume that the first component of these pairs are the same due to the first hypothesis in Lemma A.0.7).

So take any $x \in f^{-1}(y)$ and let $a \Vdash_X x$. Notice that $\mathsf{p}_2 e \Vdash s_1, s_2$, so $\mathsf{p}_2 ea \Vdash s_1(x), s_2(x)$. Also we have $\tilde{b}(s_1(x)) = y = \tilde{b}(s_2(x))$ from the definition of Π_f. Thus, as $\tilde{b}$ is a family of modest sets, again by Lemma A.0.7 we have that $s_1(x) = s_2(x)$— and so $s_1 = s_2$. □

Clearly, families of modest sets are stable under pullback along arbitrary morphisms in $\mathbf{Asm}(\mathcal{A})$. Furthermore, for any $X \in \mathbf{Asm}(\mathcal{A})$, $\mathbf{Mod}(\mathcal{A})(X)$ has finite limits, which means that, by Theorem A.0.8, $\mathbf{Mod}(\mathcal{A})(X)$ is cartesian closed. ($\mathbf{Mod}(\mathcal{A})(X)$ also has finite colimits.)

We now seek a "generic family of modest sets", ie. a family of modest sets such that all families are isomorphic to a pullback of the generic family along some morphism in $\mathbf{Asm}(\mathcal{A})$. Towards this, observe that every modest set X is isomorphic to a "canonically modest set" X_c given by:

$$\begin{aligned} |X_c| &:= \{\|x\|_X \mid x \in |X|\}, \\ \|A\|_{X_c} &:= A. \end{aligned}$$

That is, we replace every element x by its set of realizers. Then there is a one-to-one correspondence between canonically modest sets and partial equivalence relations (PERs), ie. symmetric and transitive binary relations, on $|\mathcal{A}|$. The PER corresponding to the canonically modest set X is given by

$$a R_X a' \iff \exists x \in |X|.\, a, a' \Vdash x\,.$$

On the other hand, given a PER on $|\mathcal{A}|$, we get the canonically modest set A_R with:

$$\begin{aligned} |A_R| &:= |\mathcal{A}|/R = \{[a]_R \mid aRa\}, \\ \|A\|_{A_R} &:= A. \end{aligned}$$

That is, A_R is the modest set of R-equivalence classes of $|\mathcal{A}|$, realized by their members.

Theorem A.0.9.

There exists a generic family of modest sets, that is, a family of modest sets γ such that for any family of modest sets $\tilde{b}$ there is a morphism of assemblies $\chi_{\tilde{b}}$, the "characteristic map of $\tilde{b}$", such that $\tilde{b} \cong {\chi_{\tilde{b}}}^* \gamma$.

Proof. Let $\mathsf{PER}(\mathcal{A})$ be the set of all partial equivalence relations on $|\mathcal{A}|$. Let G be the assembly defined by:

$$\begin{aligned} |G| &:= \{(R, A) \mid R \in \mathsf{PER}(\mathcal{A}) \wedge A \in |\mathcal{A}|/R\}, \\ \|(R, A)\|_G &:= A. \end{aligned}$$

The generic family of modest set is given by:

$$\begin{aligned} \gamma &: G \to \nabla(\mathsf{PER}(\mathcal{A})), \\ \gamma(R, A) &:= R. \end{aligned}$$

where ∇ is the functor from Eq. (A.0.3). γ is realized by id (remember, we can consider R as a modest set by the foregoing).

If $\tilde{b} : B \to X$ is a family of modest sets, then we have $\tilde{b} \cong {\chi_{\tilde{b}}}^* \gamma$ for the map:

$$\begin{aligned} \chi_{\tilde{b}} &: X \to \nabla(\mathsf{PER}(\mathcal{A})), \\ \chi_{\tilde{b}}(x) &:= \{(a, a') \mid \exists y \in \tilde{b}^{-1}(x).\, a, a' \Vdash_A y\}. \end{aligned}$$

□

We pause to sum up our position. We have a class of categories in the $\mathbf{Mod}(\mathcal{A})(X)$ that are cartesian closed and closed under Π_f for any morphism $f \in \mathbf{Asm}(\mathcal{A})$. Since modest families are stable under pullback, the categories $\mathbf{Mod}(\mathcal{A})(X)$ together constitute an *indexed category*, ie. a **Cat**-valued, contravariant pseudofunctor, (cf. [Cro94]):

$$\mathbf{Mod}(\mathcal{A})(-) : \mathbf{Asm}(\mathcal{A})^{\mathrm{op}} \to \mathbf{Cat}$$

Moreover, we have a generic family of modest sets, which may be expressed as a condition on the indexed category $\mathbf{Mod}(\mathcal{A})(-)$. With this we can give an interpretation of the system set out in §1.2. As already mentioned, a significant portion of the interpretation follows [See84], and so is only sound up to the usual coherence issues. Of course, there are many ways to treat these issues: see, for example: [Cur93], [Hof94], [Dyb96], [LW15] or [Awo16].

The outline of the interpretation is as follows.

- Contexts Γ are interpreted as objects $[\![\Gamma]\!]$ of $\mathbf{Asm}(\mathcal{A})$, where the empty context is interpreted as the terminal object.
- Substitutions $\sigma : \Gamma \to \Gamma'$ are interpreted as morphisms $[\![\sigma]\!] : [\![\Gamma]\!] \to [\![\Gamma']\!]$ in $\mathbf{Asm}(\mathcal{A})$.

- A type $\Gamma \vdash A$ type is interpreted as the object of $\mathbf{Asm}(\mathcal{A})/[\![\Gamma]\!]$ corresponding to the interpretation of the "projection substitution" $p_A : \Gamma, x : A \to \Gamma$:

$$[\![\Gamma \vdash A \,\mathsf{type}]\!] := [\![p_A]\!] : [\![\Gamma, x : A]\!] \to [\![\Gamma]\!]$$

  We often identify the interpretation of a closed type (an object of $\mathbf{Asm}(\mathcal{A})/1$) with its domain.
- The universe $\mathcal{U}$ type is interpreted as the assembly $\nabla(\mathsf{PER}(\mathcal{A}))$.
- The interpretation $[\![\Gamma \vdash a : A]\!]$ of a term $\Gamma \vdash a : A$ is a point of $[\![\Gamma \vdash A \,\mathsf{type}]\!]$ in $\mathbf{Asm}(\mathcal{A})/[\![\Gamma]\!]$, which is precisely a section of $[\![p_A]\!]$ in $\mathbf{Asm}(\mathcal{A})$. A small type $\Gamma \vdash B : \mathcal{U}$ is treated as a term of type $\mathcal{U}$.
- The action of substitution is given by pullback.

The interpretation of Σ-types and identity types is as usual—where the former uses the left adjoint to pullback from Theorem A.0.4. The validity of the Π-FORM$_2$ rule is shown as usual, too. We discuss the interpretation of *impredicative* Π-types, as in the Π-FORM$_1$ rule (cf. §1.2).

Given $[\![\Gamma \vdash A \,\mathsf{type}]\!]$ and $[\![\Gamma, x : A \vdash B(x) : \mathcal{U}]\!]$, we interpret the dependent function $\Gamma \vdash \prod_{(x:A)} B(x) : \mathcal{U}$ by:

$$[\![\Gamma \vdash \textstyle\prod_{(x:A)} B(x) : \mathcal{U}]\!] := \langle [\![\Gamma]\!], \chi_{\Pi_{[\![p_A]\!]}[\![\Gamma, x:A \vdash B(x) \,\mathsf{type}]\!]} \rangle : [\![\Gamma]\!] \to [\![\Gamma, X : \mathcal{U}]\!],$$

where $\chi_{\Pi_{[\![p_A]\!]}[\![\Gamma, x:A \vdash B(x) \,\mathsf{type}]\!]}$ is the characteristic map of $\Pi_{[\![p_A]\!]}[\![\Gamma, x : A \vdash B(x) \,\mathsf{type}]\!]$ (cf. Theorem A.0.9), and we are viewing $B(x)$ "as a type", that is:

$$[\![\Gamma, x : A \vdash B(x) \,\mathsf{type}]\!] := (\pi_2 \circ [\![\Gamma, x : A \vdash B(x) : \mathcal{U}]\!])^* \gamma,$$

where γ is the generic family of modest sets from Theorem A.0.9 and π_2 is as in the following pullback diagram.

$$\begin{array}{ccc}
[\![\Gamma, x : A, X : \mathcal{U}]\!] & \xrightarrow{\pi_2} & [\![\mathcal{U}]\!] \\
{\scriptstyle [\![\Gamma, x:A \vdash B(x):\mathcal{U}]\!]}\uparrow \;\downarrow {\scriptstyle [\![\Gamma, x:A \vdash \mathcal{U} \,\mathsf{type}]\!]} & \lrcorner & \downarrow {\scriptstyle [\![\vdash \mathcal{U} \,\mathsf{type}]\!]} \\
[\![\Gamma, x : A]\!] & \xrightarrow[!_{[\![\Gamma]\!]}]{} & 1
\end{array}$$

As families of modest sets are stable under pullback, so $[\![\Gamma, x : A \vdash B(x) \,\mathsf{type}]\!]$ is a modest family. This means that $\Pi_{[\![p_A]\!]}[\![\Gamma, x : A \vdash B(x) \,\mathsf{type}]\!]$ is a modest family, as modest families are preserved by $\Pi_{[\![p_A]\!]}$ (Theorem A.0.8).

Finally, observe that implicit in the above is the reason that the TP rule is validated. Given $[\![\Gamma \vdash A : \mathcal{U}]\!]$ as a section, we obtain $[\![\Gamma \vdash A \,\mathsf{type}]\!]$ as a pullback of the generic modest family (along $\pi_2 \circ [\![\Gamma \vdash A : \mathcal{U}]\!]$), which will itself be a modest family and in particular an object of $\mathbf{Asm}(\mathcal{A})/[\![\Gamma]\!]$.

Bibliography

[AGS12] Steve Awodey, Nicola Gambino, and Kristina Sojakova. Inductive types in homotopy type theory. In *Proceedings of the 2012 27th Annual IEEE/ACM Symposium on Logic in Computer Science*, pages 95–104. IEEE Computer Society, 2012. (Cited on pages 17, 18, 20, 35, and 36.)

[Awo10] Steve Awodey. *Category Theory*. Number 52 in Oxford Logic Guides. Oxford University Press, Second Edition, 2010. (Cited on page 28.)

[Awo16] Steve Awodey. Natural models of homotopy type theory. *Mathematical Structures in Computer Science*, pages 1–46, 2016. (Cited on page 47.)

[BBG+63] J. W. Backus, F. L. Bauer, J. Green, C. Katz, J. McCarthy, A. J. Perlis, H. Rutishauser, K. Samelson, B. Vauquois, J. H. Wegstein, A. van Wijngaarden, and M. Woodger. Revised report on the algorithm language algol 60. *Commun. ACM*, 6(1):1–17, January 1963. (Cited on page 1.)

[CFS88] Aurelio Carboni, Peter Freyd, and Andre Scedrov. A categorical approach to realizability and polymorphic types. *Mathematical Foundations of Programming Language Semantics*, pages 23–42, 1988. (Cited on page 41.)

[CH88] Thierry Coquand and Gérard Huet. The calculus of constructions. *Information and Computation*, 76:95–120, 1988. (Cited on page 4.)

[Cro94] Roy L. Crole. *Categories for Types*. Cambridge University Press, 1994. (Cited on pages 37 and 47.)

[Cur93] P-L Curien. Substitution up to isomorphism. *Fundamenta Informaticae*, 19(1-2):51–85, 1993. (Cited on page 47.)

[DS70] E. J. Dubuc and R. Street. Dinatural transformations. In S. MacLane, editor, *Reports of the Midwest Category Seminar IV*, volume 137 of *Lecture Notes in Mathematics*, pages 126–137. Springer-Verlag Berlin Heidelberg, 1970. (Cited on page 25.)

[Dyb96] Peter Dybjer. Internal type theory. *Types for Proofs and Programs*, pages 120–134, 1996. (Cited on page 47.)

[Fru16] Daniil Frumin. Logic and homotopy in the category of assemblies. `http://covariant.me/pdf/realiz.pdf`, 2016. (Cited on page 43.)

[Gir72] Jean-Yves Girard. *Interprétation Fonctionnelle et Elimination des Coupures de l'Arithmétique d'Ordre Supérieur*. PhD thesis, Paris 7, France, 1972. (Cited on page 1.)

[GTL89] Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and Types*. Cambridge University Press, 1989. (Cited on page 1.)

[Hof94] Martin Hofmann. On the interpretation of type theory in locally cartesian closed categories. In *Proceedings of Computer Science Logic, Lecture Notes in Computer Science*, pages 427–441. Springer, 1994. (Cited on page 47.)

[How80] William A. Howard. The formulae-as-types notion of construction. In J. Roger Seldin, Jonathan P.; Hindley, editor, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, 1980. original paper manuscript from 1969. (Cited on page 1.)

[Hyl88] J.M.E. Hyland. A small complete category. *Annals of Pure and Applied Logic*, 40(2):135 – 165, 1988. (Cited on pages 29 and 41.)

[Kle45] S. C. Kleene. On the interpretation of intuitionistic number theory. *The Journal of Symbolic Logic*, 10(4):109–124, 1945. (Cited on page 41.)

[LW15] Peter LeFanu Lumsdaine and Michael A Warren. The local universes model: an overlooked coherence construction for dependent type theories. *ACM Transactions on Computational Logic (TOCL)*, 16(3):23, 2015. (Cited on page 47.)

[Rey74] John C. Reynolds. Towards a theory of type structure. In *Programming Symposium, Proceedings Colloque sur la Programmation*, pages 408–423. Springer-Verlag, 1974. (Cited on page 1.)

[Rey83] John C. Reynolds. Types, Abstraction and Parametric Polymorphism. In *IFIP Congress*, pages 513–523, 1983. (Cited on page 4.)

[Rum04] Ivar Rummelhoff. Polynat in PER models. *Theoretical Computer Science*, 316:215–224, 2004. (Cited on pages 3 and 37.)

[See84] Robert AG Seely. Locally cartesian closed categories and type theory. In *Mathematical proceedings of the Cambridge philosophical society*, volume 95, pages 33–48. Cambridge University Press, 1984. (Cited on pages 45 and 47.)

[Str08] Thomas Streicher. Realizability. `http://www.mathematik.tu-darmstadt.de/~streicher/REAL/REAL.pdf`, 2008. (Cited on page 41.)

[SU06] Morten Heine Sørensen and Paweł Urzyczyn. *Lectures on the Curry-Howard Isomorphism*. Number 149 in Studies in Logic and the Foundations of Mathematics. Elsevier, 2006. (Cited on pages 1 and 29.)

[Uni13] Ther Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. `https://homotopytypetheory.org/book`, Institute for Advanced Study, 2013. (Cited on pages 5, 8, 10, 12, 16, 23, and 34.)

[vO08] J. van Oosten. *Realizability: An Introduction to Its Categorical Side*. Studies in logic and the foundations of mathematics. Elsevier, 2008. (Cited on page 41.)