

PRIMITIVE RECURSIVE DEPENDENT TYPE THEORY

ULRIK BUCHHOLTZ AND JOHANNES SCHIPP VON BRANITZ

ABSTRACT. We show that restricting the elimination principle of the natural numbers type in Martin-Löf Type Theory (MLTT) to a universe of types not containing Π -types ensures that all definable functions are primitive recursive. This extends the concept of primitive recursiveness to general types. We discuss extensions to univalent type theories and other notions of computability. We are inspired by earlier work by Martin Hofmann [18], work on Joyal’s arithmetic universes [25], and Hugo Herbelin and Ludovic Patey’s sketched Calculus of Primitive Recursive Constructions [16].

We define a theory T_{pr} that is a subtheory of MLTT with two universes $U_0 : U_1$, such that all inductive types are finitary and U_0 is restricted to not contain Π -types:

$$\frac{\vdash A : U_\alpha \quad a : A \vdash B(a) : U_\alpha}{\vdash (a : A) \rightarrow B(a) : U_{\max(1, \alpha)}}$$

We prove soundness such that all functions $\mathbb{N} \rightarrow \mathbb{N}$ are primitive recursive. The proof requires that T_{pr} satisfies canonicity, which we easily prove using synthetic Tait computability [33].

1. INTRODUCTION

A primitive recursive function is roughly a numerical algorithm that can be computed using only (bounded) for-loops. They form a subclass of all computable functions and are commonly used in proofs of relative consistency results.

Martin-Löf Type Theory (MLTT) is a dependent type theory which can serve as a foundation for mathematics. The variant we consider features Σ - or dependent pair and Π - or dependent function types, intensional identity types, basic inductive types and a hierarchy of universes.

The main contribution of this paper is a proof that restricting the elimination principle of the type of natural numbers to a universe not containing Π -types ensures that all definable terms $n : \mathbb{N} \vdash f(n) : \mathbb{N}$ are primitive recursive functions under their standard interpretation in the topos Set of sets. In other words, dependent type theory without Π -types is a conservative extension of primitive recursive arithmetic (PRA). The proof proceeds by gluing the set-model to a certain sheaf topos of primitive recursive functions.

PRA is often invoked as a base theory for reverse mathematics [32] and work in formal metatheory [21]. However, this line of work requires a lot of delicate encoding and would be difficult to mechanize in a proof assistant. Our work aims to alleviate this problem by giving a subsystem of MLTT (which is itself the basis for many proof assistants, including Agda, Rocq (née Coq), and Lean), which is conservative over PRA, but is much more expressive (requiring fewer encodings), and which is directly amenable for mechanization. Eventually, we imagine that a tool like Agda could feature a `--pra` flag to ensure that a file only uses constructions that are conservative over PRA. Since we provide a modular semantics that ensures this, it is easily possible to extend our syntax with even further conveniences, some of which we discuss in Section 10.

In Section 2 we recall the basic definitions of primitive recursive functions and their representation in Cartesian closed categories. In Section 3 we fix notation for the tool that is synthetic Tait computability (STC).

This is followed by a sketch of concrete syntax and a formal definition of higher order abstract syntax for our Primitive Recursive Dependent Type Theory (PRTT) in Section 4. We discuss examples and applications in Section 5.

In Section 6 we define a semantics in the topos of sheaves on a category of arities and primitive recursive functions equipped with the finite cover topology. This is followed by a proof that PRTT admits canonical forms in Section 7. The results of those two sections are finally combined

in Section 8 where we define an interpretation of PRTT in a topos, constructed using Artin gluing along the interpretations in the sheaf topos and the standard model. We use the glue topos to show that all PRTT-definable functions are in fact primitive recursive.

A comparison of our results to related work is given in Section 9.

The system PRTT has the downside that while it is complete with respect to primitive recursive functions, not every primitive recursive function can be encoded in a straightforward way. We discuss possible extensions using a comonadic modality, an internal universe of codes for primitive recursive constructions, finitary inductive types and polynomial time computability akin to Hofmann's calculi [18, 17] in Section 10. We also consider applications to Cubical Type Theory and obstructions to the transfer of our techniques to higher topoi.

2. PRIMITIVE RECURSION

In this section we define primitive recursive functions and explain why one might expect MLTT with natural numbers but without Π -types to capture exactly primitive recursive functions.

Definition 2.1. The *basic primitive recursive functions* are constant functions, the successor function and projections out of finite products of \mathbb{N} . A *primitive recursive function* is obtained by finite applications of function composition and the primitive recursion operator

$$\begin{aligned} (l : \mathbb{N}) &\rightarrow (\mathbb{N}^l \rightarrow \mathbb{N}) \rightarrow (\mathbb{N}^{l+2} \rightarrow \mathbb{N}) \rightarrow (\mathbb{N}^{l+1} \rightarrow \mathbb{N}) \\ \text{primrec}_{g,h}^l(0, x) &= g(x) \\ \text{primrec}_{g,h}^l(n+1, x) &= h(n, \text{primrec}_{g,h}^l(n, x), x) \end{aligned}$$

Many functions are primitive recursive. Some examples include addition, exponentiation and the greatest common divisor. One of the simplest and earliest-discovered examples of a total computable function which is not primitive recursive is the Ackermann function [1]. A simple two-argument variant A due to Rózsa Péter is given by

$$\begin{aligned} A(0, n) &:= n + 1 \\ A(m + 1, 0) &:= A(m, 1) \\ A(m + 1, n + 1) &:= A(m, A(m + 1, n)) \end{aligned}$$

for non-negative integers m and n . One can show that A grows faster than any primitive recursive function and is therefore not primitive recursive. The explosion in growth is caused by the third clause, featuring structural recursion targeting the function type $\mathbb{N} \rightarrow \mathbb{N}$.

We can transfer the notion of primitive recursiveness to morphisms between natural numbers objects (NNO) in general Cartesian closed categories (CCC) (c.f. [4]).

Definition 2.2. A function $f : \mathbb{N}^k \rightarrow \mathbb{N}$ is *representable* in a Cartesian category with weak NNO \mathbb{N} if there is an arrow $\tilde{f} : \mathbb{N}^k \rightarrow \mathbb{N}$ such that the following square commutes.

$$\begin{array}{ccc} \mathbb{N}^k & \xrightarrow{f} & \mathbb{N} \\ \text{num}^k \downarrow & & \downarrow \text{num} \\ \Gamma \mathbb{N}^k & \xrightarrow{\Gamma \tilde{f}} & \Gamma \mathbb{N} \end{array}$$

Here $\text{num}(n)$ denotes the n -th numeral in \mathbb{N} and $\Gamma := \text{Hom}(\mathbb{1}, -)$ the global sections functor.

The following result gives us a hint what a dependent type theory which is sound and complete w.r.t. primitive recursion might look like.

Theorem 2.3 ([20, 29, 4]). *The primitive recursive functions are exactly the representable functions in the free Cartesian category with parametrized NNO. The provably total functions of Peano Arithmetic are exactly the representable functions in the free CCC with weak NNO.*

On one hand, induction on natural numbers and products suffice to represent all primitive recursive functions. On the other hand, if the category has exponentials, more than just the primitive recursive functions are representable. In other words, when type theory is used as the internal language of such a category, one would expect the presence or absence of Π -types to determine whether just primitive recursive or all total recursive functions are definable.

3. SYNTHETIC TAIT COMPUTABILITY

In his PhD Thesis [33], Sterling introduces a synthetic method of constructing logical relations in order to prove properties such as normalisation for formal systems. The idea is that, given a left exact functor $\rho : \mathbb{T} \rightarrow \mathcal{S}$ from a theory (*qua* syntactic category) to a topos, the Artin gluing $\mathcal{G} := \hat{\rho} \downarrow \mathcal{S}$ of the extension of ρ along the free cocompletion of \mathbb{T} is a topos. This topos comes equipped with an open immersion $j : \text{Pr } \mathbb{T} \hookrightarrow \mathcal{G}$ and a closed immersion $i : \mathcal{S} \hookrightarrow \mathcal{G}$ such that $\hat{\rho} = i^* \circ j_*$. The two immersions act on sheaves as

$$\begin{aligned} j_*(X) &= (X, \text{id} : \hat{\rho}(X) \rightarrow \hat{\rho}(X)) \\ j^*(X, \varphi : S \rightarrow \hat{\rho}(X)) &= X \\ i_*(S) &= (\mathbb{1}, ! : S \rightarrow \hat{\rho}(\mathbb{1})) \\ i^*(X, \varphi : S \rightarrow \hat{\rho}(X)) &= S. \end{aligned}$$

One obtains an open modality $\circ := j_* \circ j^*$ and a closed modality $\bullet := i_* \circ i^*$ on \mathcal{G} . Given the subterminal sheaf

$$\xi := (\mathbb{1}, ! : \mathbb{0} \rightarrow \hat{\rho}(\mathbb{1})),$$

one sees that $\bullet(X, \varphi)$ is given by the pushout of the span

$$(X, \varphi) \leftarrow ((X, \varphi) \times \xi) \rightarrow \xi.$$

This yields an elimination principle for \bullet -modal types, denoted by a try-clause.

One uses the internal language of \mathcal{G} to construct a section s of the projection j^* . This section induces squares

$$\begin{array}{ccc} S & \longrightarrow & S' \\ \downarrow & & \downarrow \\ \hat{\rho}(X) & \xrightarrow{\hat{\rho}(f)} & \hat{\rho}(X') \end{array}$$

for terms $f : X \rightarrow X'$ of \mathbb{T} . The section is constructed by lifting syntactic objects, internally given by \circ -modal types, to larger universes, thereby attaching computational information to them.

In order to ensure that s is a section up to strict equality rather than isomorphism, one assumes that the relevant universes are *strong*, meaning that they satisfy *realignment* along monomorphisms. In our exposition we leave realignment mostly implicit.

We adopt Sterling's notation

$$\{A \mid \xi \hookrightarrow a\}$$

for *extent types*, that is types whose elements are strictly aligned over some ξ -partial element a .

If ρ is the global sections functor, then one can choose a section which aligns canonical forms over closed terms and hence proves canonicity for \mathbb{T} . We shall employ this technique to prove canonicity for PRTT in Section 7.

Sterling's logical framework is a dependent type theory which produces an LCCC \mathbb{T} of judgements given its specification as a total space over the judgement classifier. Such a specification is typically given using Agda-style record syntax. A complete syntax of the abstract syntax for our theory is presented in Figure 2.

Furthermore, we construct a model $\llbracket - \rrbracket_{\mathcal{R}} : \mathbb{T}_{\text{pr}} \rightarrow \mathcal{R}$ in a certain topos of sheaves on a site of primitive recursive functions and form the glue topos along the functor

$$\begin{aligned} \rho : \text{Pr } \mathbb{T}_{\text{pr}} &\rightarrow \text{Set} \\ X &\mapsto \Gamma(\widehat{\llbracket X \rrbracket_{\mathcal{R}}}) \times \widehat{\llbracket X \rrbracket_{\text{Set}}} \end{aligned}$$

(with $\widehat{\llbracket - \rrbracket_{\mathcal{R}}}$ and $\widehat{\llbracket - \rrbracket_{\text{Set}}}$ denoting the corresponding Yoneda extensions), in order to prove that the standard interpretation of a term $n : \mathbb{N} \vdash f(n) : \mathbb{N}$ in Set is indeed primitive recursive.

4. SYNTAX OF PRIMITIVE RECURSIVE DEPENDENT TYPE THEORY

We would like to restrict the induction principle of an inductive natural numbers type without entirely removing dependent products from the theory. To do so we assume a universe U_0 containing basic inductive types, which is closed under Σ -types and identity types, but not

FIGURE 1. Rules of \mathbb{T}_{pr} .

Rules which differ from standard MLTT. All judgements $\Delta \vdash \mathcal{J}$ have an implicit prepended context Γ .

$$\begin{array}{c}
\frac{}{\vdash U_\alpha \text{ type}} \quad \frac{}{\vdash A : U_\alpha} \quad \frac{}{a : A \vdash B(a) \text{ type}} \quad \frac{}{\vdash A : U_\alpha} \quad \frac{}{a : A \vdash B(a) : U_\alpha} \\
\frac{}{\vdash U_\alpha \text{ type}} \quad \frac{}{\vdash A : U_{\alpha+1}} \quad \frac{}{\vdash (a : A) \rightarrow B(a) \text{ type}} \quad \frac{}{\vdash (a : A) \rightarrow B(a) : U_{\max(1, \alpha)}} \\
\frac{}{a : A \vdash b(a) : B(a)} \quad \frac{}{\vdash f : (a : A) \rightarrow B(a)} \\
\frac{}{\vdash \lambda(a : A).b(a) : (a : A) \rightarrow B(a)} \quad \frac{}{a : A \vdash f(a) : B(a)} \\
\frac{}{x : A \vdash (\lambda(a : A).b(a))(x) = b(x) : B(x)} \quad \frac{}{\vdash \lambda(a : A).f(a) = f : (a : A) \rightarrow B(a)} \\
\frac{}{\vdash N \text{ type}} \quad \frac{}{\vdash N : U_0} \quad \frac{}{n : N \vdash \text{ind}_{c_{\text{succ}}, c_{\text{zero}}}(n) : C(n)} \\
\frac{}{\vdash \text{zero} : N} \quad \frac{}{n : N \vdash \text{succ } n : N} \quad \frac{}{n : N \vdash \text{ind}_{c_{\text{succ}}, c_{\text{zero}}}(\text{zero}) = c_{\text{zero}} : C(\text{zero})} \\
\frac{}{n : N \vdash \text{ind}_{c_{\text{succ}}, c_{\text{zero}}}(\text{succ } n) = c_{\text{succ}}(n, \text{ind}_{c_{\text{succ}}, c_{\text{zero}}}(n)) : C(\text{succ } n)}
\end{array}$$

function types. The elimination principle of the natural numbers N is restricted to type families in U_0 .

An illustrative portion of the concrete syntax for PRTT is presented in Fig. 1. Later we shall only work with its formal account as presented in Fig. 2.

A detailed account of the typical rules of MLTT is given in [27]. The only notable differences between that system and ours are that Π -formation of a family of types of universe level 0 raises the level by one, and that the induction principle for the natural numbers produces an open term $n : N \vdash \text{ind}(n) : B(n)$ rather than a function of type $(n : N) \rightarrow B(n)$. So, in addition to the rules in Figure 1, we assume the standard rules for a reflexive, symmetric and transitive judgemental equality relation and variable substitution. Additionally, we assume Σ - and identity type closure for every universe and closure under Π -types for universes U_α with $\alpha > 0$. Note that we do not assume univalence nor any extensionality principle for propositional equality. The universe levels $\alpha \leq \beta \leq \gamma$ belong to a finite or countable linear order $\{0 < 1 < \dots\}$. We denote meta-level Σ - or dependent pair types by $(a : A) \times B(a)$ and Π - or dependent function types by $(a : A) \rightarrow B(a)$. We make no distinction between a type A and its code $A : U$ in a universe, although this is of course present in the formal account.

The logical framework converts a theory presented as a nested Σ -type in Agda-style record notation into a locally Cartesian closed (LCC) category of judgements \mathbb{T}_{pr} . For example, \square denotes a judgement classifier and $\text{tm}_\alpha : \text{tp}_\alpha \rightarrow \square$ introduces for every type judgement $A : \text{tp}_\alpha$ a judgement for its terms. A model of \mathbb{T}_{pr} is an LCC functor $\llbracket - \rrbracket_{\mathbb{C}} : \mathbb{T}_{\text{pr}} \rightarrow \mathbb{C}$.

One part of the adequacy statement for \mathbb{T}_{pr} is the next theorem. Its counterpart is Theorem 8.4.

Theorem 4.1. *The theory \mathbb{T}_{pr} is complete with respect to primitive recursion in the sense that any primitive recursive function can be defined in it.*

Proof. The basic primitive recursive functions are clearly definable using ind . Function composition is given by substitution. The term ind also covers the recursor primrec^l . The extra variables are absorbed by the context. \square

Remark 4.2. The theory \mathbb{T}_{pr} is a subtheory of MLTT. As such, any model for MLTT is a model for \mathbb{T}_{pr} . This includes any presheaf topos on a small category. In particular, we have a standard model

$$\llbracket - \rrbracket_{\text{Set}} : \mathbb{T}_{\text{pr}} \rightarrow \text{Set}$$

in the topos of sets. This model maps the syntactic natural numbers N to the actual natural numbers \mathbb{N} . Below, we argue that any definable function

$$\llbracket f \rrbracket_{\text{Set}} : \text{Set}(\llbracket N \rrbracket_{\text{Set}}, \llbracket N \rrbracket_{\text{Set}})$$

is actually primitive recursive by gluing along an interpretation of \mathbb{T}_{pr} in a topos of primitive recursive functions.

Remark 4.3. The calculus \mathbb{T}_{pr} can be amended with a natural numbers type $\vdash N_\alpha : \text{tp}_\alpha$ for $\alpha > 0$ which has an elimination principle for the larger universes also closed under Π -types and which

FIGURE 2. Higher Order Abstract Syntax for T_{pr} .

Curly braces denote implicit arguments. The judgement levels $\alpha \leq \beta \leq \gamma$ range over a finite or countable linear order $\{0, \dots\}$.

record T_{pr} : SIG where

$\text{tp}_\alpha : \square$
 $\text{tm}_\alpha : \text{tp}_\alpha \rightarrow \square$
 $\langle \uparrow_\alpha^\beta \rangle : \text{tp}_\alpha \rightarrow \text{tp}_\beta$
 $- : \{A\} \rightarrow (\langle \uparrow_\alpha^\alpha \rangle A =_{\text{tp}_\alpha} A)$
 $- : \{A\} \rightarrow (\langle \uparrow_\beta^\gamma \rangle \langle \uparrow_\alpha^\beta \rangle A =_{\text{tp}_\gamma} \langle \uparrow_\alpha^\gamma \rangle A)$
 $- : \{A\} \rightarrow (\text{tm}_\alpha A =_{\square} \text{tm}_\beta (\langle \uparrow_\alpha^\beta \rangle A))$
 $- : \{A, B\} \rightarrow (\langle \uparrow_\alpha^\beta \rangle A =_{\text{tp}_\beta} \langle \uparrow_\alpha^\beta \rangle B) \rightarrow (A =_{\text{tp}_\alpha} B)$
 $\Sigma_\alpha : (A : \text{tp}_\alpha) \rightarrow (B : \text{tm}_\alpha A \rightarrow \text{tp}_\alpha) \rightarrow \text{tp}_\alpha$
 $\text{pair}_\alpha : \{A, B\} \rightarrow ((a : \text{tm}_\alpha A) \times \text{tm}_\alpha (B a)) \cong \text{tm}_\alpha (\Sigma_\alpha (A, B))$
 $- : \{A, B\} \rightarrow \langle \uparrow_\alpha^\beta \rangle \Sigma_\alpha (A, B) =_{\text{tp}_\beta} \Sigma_\beta (\langle \uparrow_\alpha^\beta \rangle A, \langle \uparrow_\alpha^\beta \rangle \circ B)$
 $\text{eq}_\alpha : (A : \text{tp}_\alpha) \rightarrow (a, b : \text{tm}_\alpha A) \rightarrow \text{tp}_\alpha$
 $\text{refl}_\alpha : \{A\} \rightarrow (a : \text{tm}_\alpha A) \rightarrow \text{eq}_\alpha (a, a)$
 $\text{eqind}_\alpha : \{A, a\} \rightarrow (P : (b : \text{tm}_\alpha A) \rightarrow (p : \text{eq}_\alpha (a, b)) \rightarrow \text{tp}_\alpha) \rightarrow \{b, p\} \rightarrow \text{tm}_\alpha (P (a, \text{refl}_\alpha (a))) \rightarrow \text{tm}_\alpha (P (b, p))$
 $- : \{A, a, P\} \rightarrow (s : \text{tm}_\alpha (P (a, \text{refl}_\alpha (a)))) \rightarrow (s =_{\text{tm}_\alpha (P (a, \text{refl}_\alpha (a)))} \text{eqind}_\alpha (P, \{a\}, \{\text{refl}_\alpha (a)\}, s))$
 $- : \{A\} \rightarrow \langle \uparrow_\alpha^\beta \rangle \text{eq}_\alpha (A) =_{\text{tp}_\beta} \text{eq}_\beta (\langle \uparrow_\alpha^\beta \rangle A)$
 $\Pi_\alpha : (A : \text{tp}_\alpha) \rightarrow (B : \text{tm}_\alpha A \rightarrow \text{tp}_\alpha) \rightarrow \text{tp}_{\max(1, \alpha)}$
 $\lambda_\alpha : \{A, B\} \rightarrow ((x : \text{tm}_{\max(1, \alpha)} (A)) \rightarrow \text{tm}_{\max(1, \alpha)} (B(x))) \cong \text{tm}_{\max(1, \alpha)} (\Pi (A, B))$
 $- : \{A, B\} \rightarrow \langle \uparrow_\alpha^\beta \rangle \Pi_\alpha (A, B) =_{\beta} \Pi_\beta (\langle \uparrow_\alpha^\beta \rangle A, \langle \uparrow_\alpha^\beta \rangle \circ B) \quad (\text{where } \alpha > 0)$
 $\emptyset : \text{tp}_0$
 $\text{exfalso} : (B : \text{tm}_0 \emptyset \rightarrow \text{tp}_0) \rightarrow (x : \text{tm}_0 \emptyset) \rightarrow \text{tm}_0 (B x)$
 $1 : \text{tp}_0$
 $\star : \text{tm}_0 1$
 $\text{unitind} : (B : \text{tm}_0 1 \rightarrow \text{tp}_0) \rightarrow (b : \text{tm}_0 (B \star)) \rightarrow (x : \text{tm}_0 1) \rightarrow \text{tm}_0 (B x)$
 $- : \{B, b\} \rightarrow (\text{unitind} (B, b, \star) =_{\text{tm}_0 (B \star)} b)$
 $N : \text{tp}_0$
 $\text{zero} : \text{tm}_0 N$
 $\text{succ} : \text{tm}_0 N \rightarrow \text{tm}_0 N$
 $\text{ind} : (B : \text{tm}_0 N \rightarrow \text{tp}_0)$
 $\rightarrow (b_{\text{zero}} : \text{tm}_0 (B \text{zero}))$
 $\rightarrow (b_{\text{succ}} : (n : \text{tm}_0 N) \rightarrow \text{tm}_0 (B n) \rightarrow \text{tm}_0 (B (\text{succ } n)))$
 $\rightarrow (x : \text{tm}_0 N) \rightarrow \text{tm}_0 (B x)$
 $- : \{B, b_{\text{zero}}, b_{\text{succ}}\} \rightarrow (\text{ind} (B, b_{\text{zero}}, b_{\text{succ}}, \text{zero}) =_{\text{tm}_0 (B \text{zero})} b_{\text{zero}})$
 $- : \{B, b_{\text{zero}}, b_{\text{succ}}, n\} \rightarrow (\text{ind} (B, b_{\text{zero}}, b_{\text{succ}}, \text{succ } n) =_{\text{tm}_0 (B (\text{succ } n))} b_{\text{succ}} (\text{ind} (B, b_{\text{zero}}, b_{\text{succ}}, n)))$
 $U_\alpha : \text{tp}_{\alpha+1}$
 $- : (\text{tm}_{\alpha+1} U_\alpha =_{\square} \text{tp}_\alpha)$

is mapped to the natural numbers object in the interpretations above. Initiality gives a coercion $N_\alpha \rightarrow \langle \uparrow_0^\alpha \rangle N$.

5. EXAMPLES AND APPLICATIONS

To illustrate how to use T_{pr} , consider the Cantor Normal Forms for ordinals less than ε , for instance as developed in [22]. First we can define a (primitive recursive) isomorphism $N \simeq 1 + N \times N$, which allows us to think of numbers as unlabeled binary trees, with a corresponding induction principle. We write 0 for the element in the left component and $\omega^\alpha + \beta$ for elements in the right component. By recursion, we define the ordering relation $<$ (with values in the booleans $1 + 1$)

such that

$$\begin{aligned} 0 &< \omega^\alpha + b \\ \alpha < \gamma &\rightarrow \omega^\alpha + \beta < \omega^\gamma + \delta \\ \beta < \delta &\rightarrow \omega^\alpha + \beta < \omega^\alpha + \delta, \end{aligned}$$

and a decidable predicate isCNF such that

$$\begin{aligned} &\text{isCNF}(0) \\ \text{isCNF}(\alpha) &\rightarrow \text{isCNF}(\beta) \rightarrow \text{left}(\beta) \leq \alpha \rightarrow \text{isCNF}(\omega^\alpha + \beta), \end{aligned}$$

where the function left gives the left subtree if it exists, else 0, and $\alpha \leq \beta := (\alpha < \beta + \alpha = \beta)$. Then T_{pr} can prove $\text{CNF} := \sum_{\alpha:\mathbb{N}} \text{isCNF}(\alpha)$ is totally ordered. Of course, by our soundness result, T_{pr} cannot prove induction along $<$ on CNF , as this would amount to induction up to ε_0 . But using the universe U_1 , this can at least be stated. If we encode the syntax of arithmetic and a proof calculus for classical first order logic, we can then prove in T_{pr} the consistency of Peano arithmetic (PA) assuming this induction principle.

Likewise, by encoding other finitary inductive type families, it is possible to encode more complicated ordinal notation systems and the syntax of more complicated foundational systems, such as type theory itself. It is then possible to define translation functions, for instance the double-negation translation from PA to Heyting's arithmetic (HA), forcing translations, realizability translations, etc. This would be even easier in an extension of T_{pr} with built-in support for such finitary inductive type families. We discuss this extension below in Section 10.

6. SEMANTICS IN A TOPOS OF PRIMITIVE RECURSIVE FUNCTIONS

This section begins with the construction of a certain sheaf topos \mathcal{R} of primitive recursive functions. The remainder of this section is dedicated to the proof of the following result.

Theorem 6.1. *There is a sound interpretation*

$$\llbracket - \rrbracket_{\mathcal{R}} : \mathsf{T}_{\text{pr}} \rightarrow \mathcal{R}$$

satisfying

$$\llbracket \mathbb{N} \rrbracket_{\mathcal{R}} = y_{\mathbb{N}}.$$

Here y is the Yoneda embedding, and \mathbb{N} is a designated object in the site, as explained below. The topos \mathcal{R} has the property that morphisms

$$\mathcal{R}(\llbracket \mathbb{N} \rrbracket_{\mathcal{R}}, \llbracket \mathbb{N} \rrbracket_{\mathcal{R}})$$

are exactly primitive recursive functions $\mathbb{N} \rightarrow \mathbb{N}$, which forms a cornerstone of our gluing argument. The sheaf $\llbracket \mathbb{N} \rrbracket_{\mathcal{R}}$ is a (non-initial) natural numbers algebra $y_{\mathbb{N}}$ and $\llbracket \mathbb{U}_0 \rrbracket_{\mathcal{R}}$ is described as a type $\hat{\mathcal{U}}_0^{\text{pr}}$ of types X with a retraction $r : y_{\mathbb{N}} \rightarrow X + 1$. To show that $\hat{\mathcal{U}}_0^{\text{pr}}$ contains the basic inductive types we need show that the sheafification of the presheaf $\mathbb{2}$ is a retract of $y_{\mathbb{N}}$. We prove soundness of the elimination principle of \mathbb{N} into X by encoding terms $x : X$ as numbers $s(x) : y_{\mathbb{N}}$ via the section s of r . The interpretation of the other type formers and higher universes is standard.

In this section we fix a primitive recursive bijection with primitive recursive inverse of type $\mathbb{N} \rightarrow \mathbb{N} \times \mathbb{N}$.

6.2. A Sheaf Topos of Primitive Recursive Functions. We begin with the definition of a Grothendieck topology on a category R of arities and primitive recursive functions.

Definition 6.3. The category R has two objects $\mathbf{1}$ and \mathbb{N} , related by morphisms $\mathsf{R}(\mathbb{N}, \mathbb{N})$, the primitive recursive functions of type $\mathbb{N} \rightarrow \mathbb{N}$, and $\mathsf{R}(\mathbf{1}, \mathbb{N}) := \mathbb{N}$. The object $\mathbf{1}$ is terminal. We sometimes write \mathbb{N}^0 instead of $\mathbf{1}$.

Lemma 6.4. *The category R has finite products. It does not have all finite limits.*

Proof. We have that $\mathbb{N} \times \mathbb{N} \cong \mathbb{N}$ in R . However, the cospan

$$\lambda_{_0} : \mathbb{N} \rightarrow \mathbb{N} \leftarrow \mathbb{N} : \lambda_{_1}$$

cannot be completed to a square in R . □

Remark 6.5. The category \mathbf{R} is equivalent to the category \mathbf{R}' with countably many objects \mathbf{N}^l , and morphisms $\mathbf{R}'(m, n)$ primitive recursive functions $\mathbf{N}^m \rightarrow \mathbf{N}^n$. This is the Lawvere theory for primitive recursion. It is analogous to the category \mathbb{P} of arities and polytime functions in [18], where it is used to proof soundness of a calculus of polytime functions.

We define a Grothendieck topology on \mathbf{R} because the topos $\mathbf{Pr} \mathbf{R}$ of presheaves on \mathbf{R} does not have the desired structure, see Remark 6.15.

Lemma 6.6. *For primitive recursive $f : \mathbf{N}^n \rightarrow \mathbf{N}$ and $g : \mathbf{N}^m \rightarrow \mathbf{N}$, the set*

$$\{f = g\} := \{(x, y) \in \mathbf{N}^n \times \mathbf{N}^m \mid fx = gy\}$$

is decidable. This means that there is a primitive recursive function

$$\chi_{\{f=g\}} : \mathbf{N}^{n+m} \rightarrow \mathbf{N}$$

such that $\chi_{\{f=g\}}(x) = 1 \Leftrightarrow x \in \{f = g\}$ and $\chi_{\{f=g\}}(x) = 0 \Leftrightarrow x \notin \{f = g\}$. Furthermore, complements and intersections of decidable subsets are decidable.

Proof. Use

$$\chi_{\{f=g\}}(x, y) = 1 - \text{sgn}(\max(fx, gy) - \min(fx, gy)),$$

where $\text{sgn} 0 = 0$ and $\text{sgn}(n+1) = 1$. □

Lemma 6.7. *Let J_{fin} be the Grothendieck topology on \mathbf{R} generated by the basis B_{fin} consisting of finite jointly surjective families. This means that a family of morphisms $\{f_i : \mathbf{N}^{n_i} \rightarrow \mathbf{N}\}$ with $n_i \in \{0, 1\}$ is basic iff it is finite and the induced map out of the coproduct in \mathbf{Set} is surjective.*

Proof. We show that J_{fin} is indeed a basis. It is clear that isomorphisms define basic covers and that families of composites of basic covering families are basic covers as well. We need to check that if $\{f_i\} \in B_{\text{fin}}(\mathbf{N})$, then for any $g : \mathbf{R}(\mathbf{N}, \mathbf{N})$ there exists a basic cover $\{h_j\} \in B_{\text{fin}}(\mathbf{N})$ such that for each j , the morphism $g \circ h_j$ factors through some f_i . Covers of $\mathbf{1}$ are trivial and singletons $\mathbf{1} \rightarrow \mathbf{N}$ can be replaced by constant maps $\mathbf{N} \rightarrow \mathbf{N}$. Let

$$S_i := \{f_i = g \wedge f_1 \neq g \wedge \dots \wedge f_{i-1} \neq g\}.$$

Then the S_i cover \mathbf{N} and are pairwise disjoint. For every finite, non-empty S_i define finitely many $h_{i,j} : \mathbf{1} \rightarrow \mathbf{N}$ picking out the elements of S_i . For every infinite S_k define an enumeration $h'_k : \mathbf{N} \rightarrow \mathbf{N}$ factoring through S_k . Clearly, for each i, k and j we have $f_i = g \circ h_{i,j}$ and $f_k = g \circ h'_k$, and the $h_{i,j}$ and h'_k together cover \mathbf{N} . □

We denote the sheaf topos as follows:

$$\mathcal{R} := \text{Sh}(\mathbf{R}, J_{\text{fin}})$$

The topology J_{fin} is subcanonical, so $y_{\mathbf{N}}$ is a sheaf and

$$\mathcal{R}(y_{\mathbf{N}}, y_{\mathbf{N}}) \cong \mathbf{R}(\mathbf{N}, \mathbf{N}).$$

Just like any Grothendieck topos, \mathcal{R} has a natural numbers object \mathbf{N} .

Theorem 6.8. *The representable sheaf $y_{\mathbf{N}}$ is not a natural numbers object in $\mathbf{Pr} \mathbf{R}$ nor in \mathcal{R} .*

Proof. There are two ways to see this. One is that, if $y_{\mathbf{N}} \cong \mathbf{N}$ were true, then the Ackermann function would be representable and hence primitive recursive.

Alternatively, assume there were a natural transformation α making the square

$$\begin{array}{ccc} y_{\mathbf{N}} & \xrightarrow{\text{succ}} & y_{\mathbf{N}} \\ \alpha \downarrow & & \downarrow \alpha \\ \mathbf{N} & \xrightarrow{+1} & \mathbf{N} \end{array}$$

commute. By the Yoneda lemma, there is a number n such that at component k , α acts as

$$\alpha_n(k)(f) = \mathbf{N}(f)(n) = \text{id}_{\mathbf{N}}(n) = n,$$

where $f : \mathbf{N}^k \rightarrow \mathbf{N}$ is primitive recursive. Chasing $k : y_{\mathbf{N}}$ along both legs of the square above, we get the contradiction

$$\alpha_n(k) + 1 = n + 1 \neq n = \alpha_n(\text{succ}(k)). \quad \square$$

6.9. Universes. Let us have a look at some candidates for $\llbracket \mathcal{U}_0 \rrbracket_{\mathcal{R}}$.

Definition 6.10. We assume strong cumulative universes $\mathcal{U}_0 < \mathcal{U}_1$ in \mathcal{R} and define the universes $\mathcal{U}_0^{\text{pr}}$, $\hat{\mathcal{U}}_0^{\text{pr}}$ and $\mathcal{U}_0^{\text{cpr}}$. We have the types (as defined in the internal language)

$$\begin{aligned} \mathcal{U}_0^{\text{pr}} &:= (X : \mathcal{U}_0) \\ &\quad \times \left((g : X) \right. \\ &\quad \quad \rightarrow (h : y_{\mathbb{N}} \rightarrow X \rightarrow X) \\ &\quad \quad \left. \rightarrow ((f : y_{\mathbb{N}} \rightarrow X) \times \text{comp}(f, g, h)) \right) \end{aligned}$$

and

$$\begin{aligned} \mathcal{U}_0^{\text{cpr}} &:= (X : \mathcal{U}_0) \times \left((\Gamma : \mathcal{U}_0) \right. \\ &\quad \rightarrow (g : \Gamma \rightarrow X) \\ &\quad \rightarrow (h : \Gamma \rightarrow y_{\mathbb{N}} \rightarrow X \rightarrow X) \\ &\quad \rightarrow ((f : \Gamma \rightarrow y_{\mathbb{N}} \rightarrow X) \\ &\quad \quad \left. \times ((\gamma : \Gamma) \rightarrow \text{comp}(f^\gamma, g^\gamma, h^\gamma))) \right) \end{aligned}$$

with

$$\begin{aligned} \text{comp}(f, g, h) &:= (f(\text{zero}) = g) \\ &\quad \times ((n : y_{\mathbb{N}}) \rightarrow (f(\text{succ } n) = h(n, fn))). \end{aligned}$$

We also have the object

$$\begin{aligned} \hat{\mathcal{U}}_0^{\text{pr}} &:= (X : \mathcal{U}_0) \times (r : y_{\mathbb{N}} \rightarrow (X + \mathbb{1})) \\ &\quad \times (s : (X + \mathbb{1}) \rightarrow y_{\mathbb{N}}) \times (r \circ s = \text{id}_{X+\mathbb{1}}) \end{aligned}$$

of types X together with a retraction $y_{\mathbb{N}} \rightarrow X + \mathbb{1}$.

The universe $\mathcal{U}_0^{\text{pr}}$ consists of the \mathcal{U}_0 -small types which \mathbb{N} can eliminate into. The variant $\mathcal{U}_0^{\text{cpr}}$ captures $y_{\mathbb{N}}$ as a parametrised natural numbers algebra by introducing an arbitrary context Γ . Ideally, we would like to put $\llbracket \mathcal{U}_0 \rrbracket_{\mathcal{R}} := \mathcal{U}_0^{\text{pr}}$, but we were unable to prove that $\mathcal{U}_0^{\text{pr}}$ is closed under Σ -types. The same problem holds for $\mathcal{U}_0^{\text{cpr}}$. Instead, we define

$$\llbracket \mathcal{U}_0 \rrbracket_{\mathcal{R}} := \hat{\mathcal{U}}_0^{\text{pr}}$$

and use coercions

$$\hat{\mathcal{U}}_0^{\text{pr}} \rightarrow \mathcal{U}_0^{\text{pr}} \leftrightarrow \mathcal{U}_0^{\text{cpr}}$$

to show that functions $y_{\mathbb{N}} \rightarrow X$ with $X : \hat{\mathcal{U}}_0^{\text{pr}}$ can be defined by induction.

Lemma 6.11. *There is a retraction $\phi : \mathcal{U}_0^{\text{cpr}} \rightarrow \mathcal{U}_0^{\text{pr}}$ forgetting the context Γ and preserving the base type X .*

Proof. Correctness of ϕ is trivial. To see that ϕ has a section, assume an elimination principle without context into X , together with terms $g : \Gamma \rightarrow X$ and $h : \Gamma \rightarrow y_{\mathbb{N}} \rightarrow X \rightarrow X$. Then we can absorb Γ into the context; for every $\gamma : \Gamma$ we get a map $f^\gamma : y_{\mathbb{N}} \rightarrow X$, i.e., an appropriate term of type $\Gamma \rightarrow y_{\mathbb{N}} \rightarrow X$. \square

6.12. Finite Types in the Universes. We show that $\hat{\mathcal{U}}_0^{\text{pr}}$ contains the sheafifications of the constant presheaves $\mathbb{0}$ and $\mathbb{1}$. The section s allows us to encode elements of X as elements of $y_{\mathbb{N}}$, while the retraction is the corresponding decoding. The summand $\mathbb{1}$ serves to allow X to be empty.

Lemma 6.13. $\mathbb{0} : \hat{\mathcal{U}}_0^{\text{pr}}$.

Proof. There is exactly one map $r_0 : y_{\mathbb{N}} \rightarrow (\mathbb{0} + \mathbb{1})$ given by $n \mapsto \text{inr } \star$. Any element of \mathbb{N} yields a section via the Yoneda embedding. \square

Lemma 6.14. *The sheafification 2^+ of 2 is a retract of $y_{\mathbb{N}}$ in \mathcal{R} . In other words, $\mathbb{1} : \hat{\mathcal{U}}_0^{\text{pr}}$.*

Proof. We first remark that $\mathbf{2}$ is separated, because no object of the site \mathbf{N} is covered by the empty family. Therefore, the $+$ -construction only needs to be applied once.

An element of $\mathbf{2}^+(\mathbf{N})$ is an equivalence class of matching families

$$\{x_f \in \mathbf{2} \mid f : \mathbf{N} \rightarrow \mathbf{N} \in P\}$$

for some cover $P \in \mathbf{J}_{\text{fin}}(\mathbf{N})$ satisfying

$$x_{f \circ g} = x_f$$

for all $g : \mathbf{N}^l \rightarrow \mathbf{N}$. Here two such matching families $\{x_f \mid f \in P\}$ and $\{y_g \mid g \in Q\}$ are equivalent when there is a common refinement $T \subset P \cap Q$ with $T \in \mathbf{J}_{\text{fin}}(\mathbf{N})$ such that $x_f = y_f$ for all $f \in T$. By a similar construction as the one used to obtain the disjoint S_i in the proof of Lemma 6.7, each matching family is completely determined by finitely many

$$\{x_{g_j}\}_{j=1}^m$$

where $\{g_j\}$ is a finite, jointly surjective, *disjoint* family of morphisms with domain and codomain \mathbf{N} . In fact, we can collect all of the g_j where $x_{g_j} = 0$, and similarly for $x_{g_j} = 1$ to obtain an equivalent matching family $\{0, 1\}$ on two morphisms.¹

The action of $\mathbf{2}^+$ on a morphism $\varphi : \mathbf{N} \rightarrow \mathbf{N}$ restricts a matching family $\{x_f\}$ to its pullback along φ . This restriction does not change the value of any individual x_f .

By the Yoneda lemma,

$$\mathcal{R}(y_{\mathbf{N}}, \mathbf{2}^+) \cong \text{Pr R}(y_{\mathbf{N}}, \mathbf{2}^+) \cong \mathbf{2}^+(\mathbf{1}).$$

Under above isomorphism, an element $u \in \mathbf{2}^+(\mathbf{1})$ is sent to the natural transformation, here denoted r_u , which at component $m \in \{0, 1\}$ is given by

$$\begin{aligned} r_u(m) : (\mathbf{N}^m \rightarrow \mathbf{N}) &\rightarrow \mathbf{2}^+(m) \\ \varphi &\mapsto \mathbf{2}^+(\varphi)(u). \end{aligned}$$

We need to find an equivalence class of matching families u , and a natural transformation s with components

$$s_m : \mathbf{2}^+(m) \rightarrow (\mathbf{N}^m \rightarrow \mathbf{N})$$

such that for all $x \in \mathbf{2}^+(m)$,

$$\mathbf{2}^+(s_m(x))(u) = x,$$

natural in m . Because sheafification is a reflective localization, s is completely determined by a map $\tilde{s} : \mathbf{2} \rightarrow y_{\mathbf{N}}$, and $\tilde{s} = s \circ \eta_{\mathbf{2}}$. By the universal property of $\mathbf{2} = \mathbb{1} +_{\text{Pr R}} \mathbb{1}$, s is determined by two global sections of $y_{\mathbf{N}}$. Let us choose the constant functions c_0 and c_1 .

For our retraction we use the cover of \mathbf{N} generated by $_ \cdot 2, _ \cdot 2 + 1 : \mathbf{N} \rightarrow \mathbf{N}$ and the matching family u with value $\mathbf{a} \in \mathbf{2}$ on $_ \cdot 2$ and \mathbf{b} on the other map. If we denote the two global sections of $\mathbf{2}$ by \mathbf{a} and \mathbf{b} , we need to show

$$\mathbf{2}^+(c_0)(u) = \mathbf{a} \quad \text{and} \quad \mathbf{2}^+(c_1)(u) = \mathbf{b}.$$

It is easy to see that the pullback $c_n^*(S)$ of any \mathbf{J}_{fin} -sieve S on \mathbf{N} along a constant function $c_n : \mathbf{N} \rightarrow \mathbf{N}$ is the maximal sieve on \mathbf{N} . Since the restriction maps $\mathbf{2}^+(f)$ are locally (on each component of the equivalence class of) the matching family given by identity functions, it follows that the desired equations hold. \square

Remark 6.15. The reason we work in the topos \mathcal{R} rather than Pr R is that $\mathbf{2}$ is not a retract of $y_{\mathbf{N}}$ in Pr R , because retracts of representables in presheaf categories are tiny, but the endofunctor $X \mapsto X \times X$ on Pr R does not preserve colimits.

¹That is true if not all sections are 0, or 1. Otherwise, we get a singleton matching family $\{0\}$ or $\{1\}$.

6.16. **$y_{\mathbf{N}}$ -elimination.** In this subsection we show that $y_{\mathbf{N}}$ -retracts satisfy the elimination principle ind in several steps.

Lemma 6.17. *The representable sheaf $y_{\mathbf{N}}$ is in $\mathcal{U}_0^{\text{pr}}$.*

Proof. We show the result for the equivalent category \mathbf{R}' from Remark 6.5. Note that for any $F : \text{Pr } \mathbf{R}$ and $l \in \mathbf{N}$,

$$(y_{\mathbf{N}} \Rightarrow F)(\mathbf{N}^l) \cong \text{Pr } \mathbf{R}(y_{\mathbf{N}} \times y_{\mathbf{N}^l}, F) \cong F(\mathbf{N}^{l+1}).$$

We define the presheaf F^+ by $F^+(\mathbf{N}^l) := F(\mathbf{N}^{l+1})$. It follows that

$$\begin{aligned} & \text{Pr } \mathbf{R}(y_{\mathbf{1}}, y_{\mathbf{N}} \rightarrow (y_{\mathbf{N}} \rightarrow y_{\mathbf{N}} \rightarrow y_{\mathbf{N}}) \rightarrow y_{\mathbf{N}} \rightarrow y_{\mathbf{N}}) \\ & \cong \text{Pr } \mathbf{R}(y_{\mathbf{N}} \times y_{\mathbf{N}}^{++} \times y_{\mathbf{N}}, y_{\mathbf{N}}) \end{aligned}$$

We can define such a natural transformation which at component l is of type

$$(\mathbf{N}^l \rightarrow \mathbf{N}) \times (\mathbf{N}^{l+2} \rightarrow \mathbf{N}) \times (\mathbf{N}^l \rightarrow \mathbf{N}) \rightarrow (\mathbf{N}^l \rightarrow \mathbf{N})$$

and given by

$$(g, h, n) \mapsto \lambda x. \text{primrec}_{g,h}^l(n(x), x).$$

The computation rules are easy to verify. □

Lemma 6.18. *The sheaf 2^+ satisfies the contextual elimination principle, i.e., $2^+ : \mathcal{U}_0^{\text{cpr}}$.*

Proof. By Lemma 6.11 it suffices to show that $2^+ : \mathcal{U}_0^{\text{pr}}$. Let $(2^+, r, s, p)$ be a retraction as constructed in Lemma 6.14, $g : 2^+$, and $h : y_{\mathbf{N}} \rightarrow 2^+ \rightarrow 2^+$. We define

$$\tilde{g} := s(g) : y_{\mathbf{N}}$$

and

$$\begin{aligned} \tilde{h} & : y_{\mathbf{N}} \rightarrow y_{\mathbf{N}} \rightarrow y_{\mathbf{N}} \\ \tilde{h}(n, x) & := sh(n, rx). \end{aligned}$$

By $y_{\mathbf{N}}$ -induction (c.f. Lemma 6.17) we get

$$\begin{aligned} \tilde{f} & : y_{\mathbf{N}} \rightarrow y_{\mathbf{N}} \\ \tilde{f}(0) & = s(g) \\ \tilde{f}(\text{succ}(n)) & = sh(n, r\tilde{f}x). \end{aligned}$$

We define the desired morphism $f := r \circ \tilde{f}$. Then

$$f(0) = rsg = g$$

and

$$f(\text{succ } n) = rsh(n, r\tilde{f}n) = h(n, fn)$$

for all $n : y_{\mathbf{N}}$. □

Remark 6.19. The construction of Lemma 6.18 works for any type X with retraction $y_{\mathbf{N}} \rightarrow X$. The difficult part is proving that a retraction $y_{\mathbf{N}} \rightarrow X + \mathbb{1}$ gives $X : \mathcal{U}_0^{\text{pr}}$, and contextual elimination into 2^+ suffices to prove that, see below.

Lemma 6.20. *The predecessor function y_{pred} is a section of the successor function viewed as*

$$\text{succ} : y_{\mathbf{N}} \rightarrow (n : y_{\mathbf{N}}) \times (n \neq 0).$$

This is true in $\text{Pr } \mathbf{R}$ and \mathcal{R} .

Proof. We prove the result for $\text{Pr } \mathbf{R}$ first. We cannot yet use induction on $y_{\mathbf{N}}$. Let $n : Z \rightarrow y_{\mathbf{N}}$ such that $n \neq 0$. We show that $\text{succ}(y_{\text{pred}} n) = n$ so that function extensionality yields the result. We may assume that Z is representable (cf. [23, III.6 and VI.7]). Then the result follows immediately from the Yoneda lemma.

Because the J_{fin} is subcanonical, application of sheaffication to above identity implies the result for \mathcal{R} . □

Lemma 6.21. $y_{\mathbf{N}} : \hat{\mathcal{U}}_0^{\text{pr}}$

Proof. We inductively define

$$\begin{aligned} s : y_{\mathbb{N}} + \mathbb{1} &\rightarrow y_{\mathbb{N}} \\ n &\mapsto \text{succ } n \\ \star &\mapsto 0. \end{aligned}$$

By case distinction we get a map

$$\begin{aligned} r : y_{\mathbb{N}} &\rightarrow y_{\mathbb{N}} + \mathbb{1} \\ n &\mapsto \begin{cases} \star, & (n = 0) \\ y_{\text{pred}} n, & (n \neq 0). \end{cases} \end{aligned}$$

There is no induction needed to verify that s is a section of r . \square

Lemma 6.22. $y_{\mathbb{N}}$ has decidable equality in \mathcal{R} .

Proof. It suffices to decide $n =_{y_{\mathbb{N}}} 0$. We inductively define (cf. Lemma 6.18)

$$\begin{aligned} \pi : y_{\mathbb{N}} &\rightarrow \mathbb{2} \\ 0 &\mapsto 0 \\ \text{succ } n &\mapsto 1 \end{aligned}$$

and

$$\begin{aligned} \varphi : y_{\mathbb{N}} &\rightarrow \mathbb{2} \rightarrow y_{\mathbb{N}} \\ \varphi_n(0) &:= 0 \\ \varphi_n(1) &:= n. \end{aligned}$$

Decidability of $\pi(n) =_2 0$ and the identities

$$\varphi_n(\pi 0) = 0 \quad \text{and} \quad \varphi_n(\pi n) = n$$

can be used to decide $n =_{y_{\mathbb{N}}} 0$. \square

Theorem 6.23. *There is a map*

$$\Phi : \hat{\mathcal{U}}_0^{\text{pr}} \rightarrow \mathcal{U}_0^{\text{pr}}$$

preserving underlying types.

Proof. Let $(X, r, s, p) : \hat{\mathcal{U}}_0^{\text{pr}}$, $g : X$ and $h : y_{\mathbb{N}} \rightarrow X \rightarrow X$. We inductively (Lemma 6.17) define a helper function $\tilde{f} : y_{\mathbb{N}} \rightarrow y_{\mathbb{N}}$ with initial value

$$\tilde{g} := sg$$

and step function

$$\begin{aligned} \tilde{h} : y_{\mathbb{N}} &\rightarrow y_{\mathbb{N}} \rightarrow y_{\mathbb{N}} \\ \tilde{h}(n, x) &:= \begin{cases} sh(n, rx), & (rx : X) \\ sg, & (rx = \star). \end{cases} \end{aligned}$$

Then, we define

$$\begin{aligned} f : y_{\mathbb{N}} &\rightarrow X \\ f(n) &:= \begin{cases} r\tilde{f}n, & (r\tilde{f}n : X) \\ g, & (r\tilde{f}n = \star). \end{cases} \end{aligned}$$

We verify the computation rules of this f . We clearly have that $f(0) = rsg = g$. It is also true that

$$f(\text{succ } n) = r(\tilde{f}(\text{succ } n)) = rsh(n, r\tilde{f}n) = h(n, f n),$$

if $r(\tilde{f}(\text{succ } n)) : X$. Otherwise, this equality does not necessarily hold, so we need to show that $(n : y_{\mathbb{N}}) \rightarrow (x : X) \times (r\tilde{f}(\text{succ } n) = x)$. This x is of course given by $h(n, r(\tilde{f} n))$, but we don't have a suitable induction principle to show this. Instead, we prove that there is a lift

$$\begin{array}{ccc} & & X \\ & \nearrow \text{dashed} & \downarrow \text{inl} \\ y_{\mathbb{N}} & \xrightarrow{r \circ \tilde{f}} & X + \mathbb{1}. \end{array}$$

Since

$$\begin{array}{ccc} X & \xrightarrow{!} & \mathbb{1} \\ \text{inl} \downarrow & & \downarrow \text{inl} \\ X + \mathbb{1} & \xrightarrow{\langle !_X, !_1 \rangle} & \mathbb{1} + \mathbb{1} \end{array}$$

is a pullback square, it suffices to show that for any $n : Z \rightarrow y_{\mathbb{N}}$ there is a lift

$$\begin{array}{ccc} & & \mathbb{1} \\ & \nearrow \text{dashed} & \downarrow \text{inl} \\ Z & \xrightarrow{\langle !_X, !_1 \rangle r \tilde{f} n} & \mathbb{1} + \mathbb{1}. \end{array}$$

Switching back to type theoretic language, we have to show that

$$\langle !_X, !_1 \rangle r(\tilde{f} n) = \text{inl} \star.$$

If $n = 0$, then

$$\langle !_X, !_1 \rangle r(\tilde{f} 0) = \langle !_X, !_1 \rangle r s g = \langle !_X, !_1 \rangle g = \text{inl} \star.$$

Otherwise, $n = \text{succ}(y_{\text{pred}} n)$, and we compute

$$\langle !_X, !_1 \rangle r \tilde{f}(\text{succ}(y_{\text{pred}} n)) = \langle !_X, !_1 \rangle r \tilde{h}(n, \tilde{f}(\text{succ}(y_{\text{pred}} n))).$$

There we need to make a case distinction for whether

$$r \tilde{f}(\text{succ}(y_{\text{pred}} n)) : X.$$

However, in both cases the right hand side reduces to $\text{inl} \star$. \square

Theorem 6.23 gives us the non-dependent elimination principle for $y_{\mathbb{N}}$. The dependent one for a family of types in $\hat{\mathcal{U}}_0^{\text{pr}}$ can be deduced by taking the Σ -type, as we explain below.

6.24. Closure Under Σ - and Identity Types. In this section we show that the interpretation of Σ - and identity types is sound.

Lemma 6.25. *The universe $\hat{\mathcal{U}}_0^{\text{pr}}$ is closed under Σ -types.*

Proof. Assume we have $\bar{A} = (A, r, s, p) : \hat{\mathcal{U}}_0^{\text{pr}}$, and $\bar{B}(a) = (B(a), r_a, s_a, p_a) : \hat{\mathcal{U}}_0^{\text{pr}}$ given $a : A$. Then

$$\begin{aligned} y_{\mathbb{N}} \times y_{\mathbb{N}} &\rightarrow (a : A) \times B(a) \\ (m, n) &\mapsto (r m, r_{r m} n) \end{aligned}$$

has right inverse the map

$$\begin{aligned} (a : A) \times B(a) &\rightarrow y_{\mathbb{N}} \times y_{\mathbb{N}} \\ (a, b) &\mapsto (s a, s_a b). \end{aligned}$$

We have a composite retraction

$$y_{\mathbb{N}} \rightarrow y_{\mathbb{N}} \times y_{\mathbb{N}} \rightarrow (a : A) \times B(a). \quad \square$$

Lemma 6.26. *$\hat{\mathcal{U}}_0^{\text{pr}}$ is closed under identity types of terms.*

Proof. Assume we have $\bar{X} = (X, r, s, p) : \hat{\mathcal{U}}_0^{\text{pr}}$ and $x, y : X$. We shall define a retraction $r' : y_{\mathbb{N}} \rightarrow (x =_X y) + \mathbb{1}$ with section s' . By application of r , the type $(x =_X y) + \mathbb{1}$ is a retract of $(s(x) =_{y_{\mathbb{N}}} s(y)) + \mathbb{1}$. If $s(x) =_{y_{\mathbb{N}}} s(y)$, then $(s(x) =_{y_{\mathbb{N}}} s(y)) + \mathbb{1} \simeq \mathbb{2}$. Otherwise, $(s(x) =_{y_{\mathbb{N}}} s(y)) + \mathbb{1} \simeq \mathbb{1}$. In both cases we have a retraction r' of the desired type. \square

FIGURE 3. Lifted Syntax for Canonicity

$$\begin{aligned}
\mathbb{N}^* &: \{\text{tp}^* \mid \xi \hookrightarrow \mathbb{N}\} \\
\mathbb{N}^* &:= (\mathbb{N}, \Sigma_{\text{syn}: \text{tm}_0 \mathbb{N}} \bullet (\Sigma_{n: \mathbb{N}} (\text{syn} = \text{succ}^n \text{zero}))) \\
\text{zero}^* &: \{\text{tm}^*(\mathbb{N}^*) \mid \xi \hookrightarrow (\text{zero}, \star)\} \\
\text{zero}^* &:= (\text{zero}, \eta(0, \star)) \\
\text{succ}^* &: \{\text{tm}^*(\mathbb{N}^*) \rightarrow \text{tm}^*(\mathbb{N}^*) \mid \xi \hookrightarrow (\text{succ}, \lambda _ . \star)\} \\
\text{succ}^*(\text{syn}, \text{sem}) &: \{\Sigma_{\text{syn}: \text{tm}_0 \mathbb{N}} \bullet (\Sigma_{n: \mathbb{N}} (\text{syn} = \text{succ}^n \text{zero})) \mid \xi \hookrightarrow (\text{succ}(\text{syn}), \star)\} \\
\text{succ}^*(\text{syn}, \text{sem}) &:= \text{try sem } [\alpha \mid \xi \hookrightarrow (\text{succ}(\text{syn}), \star)] \\
&\quad \text{where} \\
&\quad \alpha : \Sigma_{n: \mathbb{N}} (\text{syn} = \text{succ}^n \text{zero}) \\
&\quad \quad \rightarrow \{\Sigma_{\text{syn}': \text{tm}_0 \mathbb{N}} \bullet \Sigma_{n: \mathbb{N}} (\text{syn}' = \text{succ}^{n+1} \text{zero}) \mid \xi \hookrightarrow (\text{succ}(\text{syn}), \star)\} \\
&\quad \alpha(n, p) = (\text{succ}(\text{syn}), \eta(n+1, \text{app}_{\text{succ}}(p))) \\
\text{ind}^* &: \{(C : \mathbb{N}^* \rightarrow \text{tp}^*) \\
&\quad \rightarrow (c_{\text{zero}} : C(\text{zero})) \\
&\quad \rightarrow (c_{\text{succ}} : (n : \mathbb{N}^*) \rightarrow C(n) \rightarrow C(\text{succ } n)) \\
&\quad \rightarrow (\text{syn} : \mathbb{N}^*) \\
&\quad \rightarrow C(\text{syn}) \mid \xi \hookrightarrow \text{ind}\} \\
\text{ind}^*(C, c_{\text{zero}}, c_{\text{succ}}, x) &:= \text{try syn.sem}[\beta \mid \xi \hookrightarrow \text{ind}(C, c_{\text{zero}}, c_{\text{succ}}, \text{syn})] \\
&\quad \text{where} \\
&\quad \beta : \Sigma_{n: \mathbb{N}} (\text{syn} = \text{succ}^n \text{zero}) \rightarrow \{C(\text{syn}) \mid \xi \hookrightarrow \text{ind}(C, c_{\text{zero}}, c_{\text{succ}}, \text{syn})\} \\
&\quad \beta := \begin{cases} (0, _) & \mapsto c_{\text{zero}} \\ (m+1, _) & \mapsto c_{\text{succ}}(\text{succ}^m \text{zero}, \beta(m, \text{refl})) \end{cases}
\end{aligned}$$

Since $(x =_X y)$ denotes strict equality in \mathcal{R} , identity induction and its computation rules hold as well.

7. CANONICITY

In this section we use synthetic Tait computability to prove canonicity for \mathbb{T}_{pr} . We remark that this result does not immediately follow from [33, § 4.5.3], because the type theory presented there does not contain a type of natural numbers. Other than that, the strategy of the proof is exactly analogous in that we lift the syntax from a strong \circ -modal universe to a larger, strong one. Since liftings of ‘negative’ types such as Π -, Σ - and identity types is trivial, we shall not repeat their definitions here.

Theorem 7.1. *Let $m : 1 \rightarrow \text{tm}_0(\mathbb{N}) : \mathbb{T}_{\text{pr}}$ be a closed term of natural numbers type; then there exists an $n : \mathbb{N}$ such that $m = \text{succ}^n \text{zero}$, meant as a statement about global elements in Set .*

Proof. Form the glue topos \mathcal{G} along the left exact global sections functor $\Gamma : \text{Pr } \mathbb{T}_{\text{pr}} \rightarrow \text{Set}$. Using the natural numbers object \mathbb{N} of \mathcal{G} , we lift \mathbb{N} , zero , succ and ind . The computation rules for ind^* follow from the ones for ind .

We remark that the partial elements $(\text{succ syn}, \star)$ appearing in the extent types and try statements have an implicit $\lambda z : \xi$ in front. The definition of α is valid, because

$$z : \xi \vdash \eta_{\lambda \xi}(n+1, \text{app}_{\text{succ}}(p)) = \star : \bullet(\dots).$$

A term $m : \mathbb{1}_{\text{Pr T}_{\text{pr}}} \rightarrow \text{tm}_0 \mathbb{N}$ lifts to $m^* : \{\mathbb{1}_{\mathcal{G}} \rightarrow \text{tm}^*(\mathbb{N}^*) \mid \xi \hookrightarrow m\}$. Unfolding the definition, we have a global element of $\bullet(\Sigma_{n:\mathbb{N}}(m = \text{succ}^n \text{zero}))$. This computes as follows.

$$\begin{aligned}
& \text{Hom}_{\mathcal{G}}(1_{\mathcal{G}}, \bullet(\Sigma_{n:\mathbb{N}}(m = \text{succ}^n \text{zero}))) \\
&= \text{Hom}_{\mathcal{G}}(1_{\mathcal{G}}, i_* i^*(\Sigma_{n:\mathbb{N}}(m = \text{succ}^n \text{zero}))) && \text{(by definition)} \\
&\cong \text{Hom}_{\text{Set}}(1_{\text{Set}}, i^*(\Sigma_{n:\mathbb{N}}(m = \text{succ}^n \text{zero}))) && (i_* \text{ lex}) \\
&\cong \text{Hom}_{\text{Set}}(1_{\text{Set}}, \Sigma_{n:\mathbb{N}} i^*(m = \text{succ}^n \text{zero})) && (i^* \text{ cocont.}) \\
&\cong \text{Hom}_{\text{Set}}(1_{\text{Set}}, \Sigma_{n:\mathbb{N}}(i^*(m) = i^*(\text{succ}^n \text{zero}))) && (i^* \text{ lex}) \\
&= \text{Hom}_{\text{Set}}(1_{\text{Set}}, \Sigma_{n:\mathbb{N}}(m = \text{succ}^n \text{zero}))
\end{aligned}$$

The last step is true, because m above is actually $j_*(y_m)$ and so $i^*(j_*(y_m)) = \Gamma y_m = m$. \square

8. SOUNDNESS FOR PRIMITIVE RECURSION

In this section we use a synthetic gluing argument to show that the set-theoretical interpretation of functions actually coincides with the primitive recursive one. The interpretations $\llbracket - \rrbracket_{\text{Set}}$ and $\llbracket - \rrbracket_{\mathcal{R}}$ are LCC functors. Hence, they extend to left exact functors $\widehat{\llbracket - \rrbracket}_{\mathcal{R}}$ and $\widehat{\llbracket - \rrbracket}_{\text{Set}}$ along their Yoneda embedding. It follows that we have a left exact extension

$$\rho := \Gamma(\widehat{\llbracket - \rrbracket}_{\mathcal{R}}) \times \widehat{\llbracket - \rrbracket}_{\text{Set}} : \text{Pr T}_{\text{pr}} \rightarrow \text{Set}$$

and the comma category $\mathcal{G} := \text{Set} \downarrow \rho$ is a topos.

Definition 8.1. For the universe $\bar{U}_0 : \dot{U}_0 \rightarrow U_0$ in T_{pr} we define the morphism

$$\llbracket \bar{U}_0 \rrbracket_{\mathcal{G}} : \llbracket \dot{U}_0 \rrbracket_{\mathcal{G}} \rightarrow \llbracket U_0 \rrbracket_{\mathcal{G}}$$

in \mathcal{G} by the square

$$\begin{array}{ccc}
\Gamma(\dot{U}_0) & \xrightarrow{\Gamma(\bar{U})} & \Gamma(U_0) \\
\llbracket \dot{U}_0 \rrbracket_{\mathcal{G}} \downarrow & & \downarrow \llbracket U_0 \rrbracket_{\mathcal{G}} \\
\rho(\dot{U}_0) & \xrightarrow{\rho(\bar{U})} & \rho(U_0),
\end{array}$$

where the vertical arrows are each given by the ‘diagonal’

$$x \mapsto (\Gamma(\llbracket x \rrbracket_{\mathcal{R}}), \llbracket x \rrbracket_{\text{Set}}).$$

Lemma 8.2. *The map $\llbracket \bar{U}_0 \rrbracket_{\mathcal{G}}$ is a universe for objects $\Gamma(X) \rightarrow \rho(X)$ given by the same diagonal as above.*

Proof. A pullback square

$$\begin{array}{ccc}
\dot{X} & \longrightarrow & \dot{U}_0 \\
\downarrow & & \downarrow \bar{U} \\
X & \longrightarrow & U_0,
\end{array}$$

induces a square

$$\begin{array}{ccc}
\Gamma(\dot{X}) & \longrightarrow & \Gamma(\bar{U}) \\
\downarrow & & \downarrow \llbracket \bar{U}_0 \rrbracket_{\mathcal{G}} \\
\rho(X) & \longrightarrow & \rho(U_0)
\end{array}$$

in \mathcal{G} . Because ρ and Γ preserve pullbacks, the top and bottom sides of the corresponding cube of sets are pullbacks, too. It follows that the square is a pullback in \mathcal{G} as well. Realignment follows from realignment of the universes in \mathcal{R} and Set . \square

Theorem 8.3. *There is a sound interpretation $\llbracket - \rrbracket_{\mathcal{G}} : \text{T}_{\text{pr}} \rightarrow \mathcal{G}$.*

Proof. The interpretation of the type and term judgements needs to be defined first. In [33, Section 4.4], Sterling defines a general way of lifting a syntactic model, that is an object of $\circ\llbracket U_0 \rrbracket_{\mathcal{G}}$ to a computability model along a hierarchy of strong universes $\llbracket U_0 \rrbracket_{\mathcal{G}} \leq \mathcal{V} \leq \mathcal{W}$. The larger universes are assumed to be closed under Π -types.

We use the same lifting of type and term judgements and refrain from repeating it here, to keep the focus on the recursion principle.

The interpretation $\llbracket U_0 \rrbracket_{\mathcal{G}}$ of the universe U_0 is defined in the previous lemma. It is clearly aligned over U_0 . It inherits its closure under Σ - and identity types. Since U_0 is not closed under Π -types, it is not an issue that Γ might not preserve them.

We put

$$\llbracket N \rrbracket_{\mathcal{G}} : \Gamma(N) \rightarrow \rho(N)$$

to be the same ‘diagonal’ as above, aligned over N .

By soundness of the interpretations $\llbracket - \rrbracket_{\mathcal{R}}$ and $\llbracket - \rrbracket_{\text{Set}}$, any syntactic term $f : N \rightarrow X$ yields a filler

$$\begin{array}{ccc} \Gamma(N) & \xrightarrow{\Gamma(f)} & \Gamma(X) \\ \llbracket N \rrbracket_{\mathcal{G}} \downarrow & & \downarrow \llbracket X \rrbracket_{\mathcal{G}} \\ \rho(N) & \xrightarrow{\rho(f)} & \rho(X). \end{array}$$

This shows that the obvious interpretations of succ and non-dependent ind are sound as well. The dependent elimination principle for N follows from the non-dependent one by eliminating into and projecting from the total space of the type family.

The hierarchy of universes and Π -types can be lifted in the same way as described in [33]. \square

Theorem 8.4. *All T_{pr} -definable functions between the natural numbers are primitive recursive. To be precise, for any term*

$$n : N \vdash f(n) : N$$

in T_{pr} the function $\llbracket \lambda(n : N).f(n) \rrbracket_{\text{Set}}$ is primitive recursive.

Proof. We made sure that the model $\llbracket - \rrbracket_{\mathcal{G}}$ is aligned over T_{pr} . In other words, we have defined a structure-preserving section of the projection $\mathcal{G} \rightarrow \text{Pr } T_{\text{pr}}$. Hence, the assumed closed term is interpreted into the computability algebra as a global element

$$f^* : 1_{\mathcal{G}} \rightarrow \{\text{tm}^*(N \rightarrow N)^* \mid \xi \hookrightarrow \lambda(n : N).f(n)\}.$$

Unfolding the definitions, we get a commutative square

$$\begin{array}{ccc} \Gamma(N) & \xrightarrow{\Gamma(f)} & \Gamma(N) \\ \llbracket N \rrbracket_{\mathcal{G}} \downarrow & & \downarrow \llbracket N \rrbracket_{\mathcal{G}} \\ \Gamma(y_1) \times \mathbf{N} & \xrightarrow{\Gamma(\llbracket f \rrbracket_{\mathcal{R}}) \times \llbracket f \rrbracket_{\text{Set}}} & \Gamma(y_1) \times \mathbf{N}. \end{array}$$

By canonicity, we have that $\Gamma(N) \cong \mathbf{N}$. It is also true that $\Gamma(y_{\mathbf{N}}) \cong \mathbf{N}$. This implies commutativity of the diagram

$$\begin{array}{ccc} \mathbf{N} & \xrightarrow{\llbracket f \rrbracket_{\text{Set}}} & \mathbf{N} \\ \wr \downarrow & & \uparrow \wr \\ \Gamma(y_{\mathbf{N}}) & \xrightarrow{\Gamma(\llbracket f \rrbracket_{\mathcal{R}})} & \Gamma(y_{\mathbf{N}}). \end{array}$$

Here $\Gamma(\llbracket f \rrbracket_{\mathcal{R}})$ is primitive recursive. In other words, the set-theoretic interpretation of f is in fact a primitive recursive function. \square

Remark 8.5. Usually, when proving theorems about all open terms, such as normalisation, one has to use a *figure shape* to determine which judgements are contexts. In the previous theorem we were able to avoid a more complex gluing situation by interpreting the λ -abstraction of f and converting the global section of the exponential back to a morphism.

9. RELATED WORK

The novel contribution of this work is the conservativity of primitive recursion with respect to dependent types with a full proof of soundness and canonicity.

Previously, Herbelin and Patey sketched a similar system [16], the Calculus of Primitive Recursive Constructions. It is a subsystem of the Calculus of Inductive Constructions which is also conservative over Primitive Recursive Arithmetic. The proposed soundness proof is similar in spirit, encoding types as primitive recursively decidable predicates on \mathbb{N} . However, this simple soundness proof doesn't work with extensions to types in U_1 , where we have function types, which are crucial for expressivity (even if they cannot be the target of inductions). Our system leverages the full power of dependent type theory, and is also closer to the syntax used in proof assistants such as Agda.

An extension of MLTT by additional recursion operators for well-founded relations has been studied by Paulson (c.f. [26]). In recursion theory one often considers partial recursive functions. These have been studied in the context of type theory as well. In [6, 9, 8, 10] the authors use an inductive domain predicate that characterises the inputs on which a function terminates. Alternative approaches such as [7, 11] associate to each data type a coinductive type of partial elements. Variants of these systems have been designed for use in proof assistants (c.f. [13]).

Our system T_{pr} is an extension of simply typed lambda calculus to full dependent type theory, potentially enhanced using a comonadic modality. One important intermediate system is the modal lambda calculus defined in [14] in order to give a formal account of *staged computation* and it is a precursor to [17].

10. CONCLUSION AND FUTURE WORK

We have shown that constructions in dependent type theory that use elimination out of the natural numbers into universes without Π -types are primitive recursive in nature. We discuss how the system can be further conservatively extended.

A benefit of our semantic approach is that extensions can be modularly added as long as they can be interpreted in \mathcal{R} and \mathcal{G} . The only drawback is that such interpretations are rather involved and use a mixture of dependent type theory (in the internal language) mixed with external reasoning about sheaves.

We expect that finitary inductive types such as lists, as well as finitary inductive type families and even small finitary induction-recursion, can be conservatively added to the calculus. This will greatly facilitate the practical mechanization of metatheory.

One could add another primitive recursive universe $\hat{U}_0 : \text{tp}_0$ with $\text{tm}_0 \hat{U}_0 = \text{tm}_0 U_0$, yielding a code $u_0 : \text{tm}_0(U_0)$ for all types tp_0 . One does not run into Girard's paradox due to the lack of function types. For this, however, we need to go beyond retracts of $y_{\mathbb{N}}$, to something like Σ_1^0 -definable types in \mathcal{R} . A weaker version of this primitive recursive universe of codes is already definable internally by using a primitive recursive Gödel encoding of the codes $\text{tm}_0 U_0$. However, its usefulness is uncertain because its codes only give tp_0 -types up to equivalence rather than judgemental equality.

It might be possible to overcome the inconvenience of not being able to define certain functions out of the natural numbers in a natural way by adding a comonadic modality \square to the theory, for example using the framework MTT [15] with mode theory the walking adjunction or the walking comonad. The new type $\square\mathbb{N}$ will have a general induction principle and it should be possible to define terms $m, n : \mathbb{N} \vdash \hat{f}(m, n) : \mathbb{N}$ given $m : \square\mathbb{N}, n : \mathbb{N} \vdash f(m, n) : \mathbb{N}$ and $m : \mathbb{N}, n : \square\mathbb{N} \vdash g(m, n) : \mathbb{N}$ defined by simultaneous induction together with an extensionality proof $(m, n : \mathbb{N}) \vdash f(\eta m, n) = g(m, \eta n)$.

A semantics playing the role of $\llbracket - \rrbracket_{\mathcal{R}}$, which we found most likely to be suitable, is as follows. Let \mathcal{S} denote the category of arities and functions between powers of natural numbers. There is a category $\mathcal{S} \rtimes \mathcal{R}$ with objects pairs of natural numbers and morphisms $(\vec{u}, \vec{v}) : (\mathcal{S} \rtimes \mathcal{R})(m, n), (m', n')$ consisting of m' set-theoretic functions $u_i : \mathbb{N}^m \rightarrow \mathbb{N}$ and n' set-theoretic functions $v_j : \mathbb{N}^{m+n} \rightarrow \mathbb{N}$ such that for each $x : \mathbb{N}^m$ the slice $v_j(x, -)$ is primitive recursive. The projection $\pi : (\mathcal{S} \rtimes \mathcal{R}) \rightarrow \mathcal{S}$ has a right adjoint inclusion functor $(-, 0)$, which is also left adjoint to the functor $+$: $(\vec{u}, \vec{v}) \mapsto (\vec{u}\vec{v}, \cdot)$. The induced string of adjunctions on presheaf categories

gives the required dependent right adjoint comonadic modality $+^* \circ (-, 0)^*$ (c.f. [15, Lemma 8.2, Theorem 7.1]). It is yet to be checked whether N can be soundly interpreted as $y_{(0,1)}$ in this setting, or if changes to the base categories, sheafification or passing to MATT [31] are needed.

The category $S \times R$ is reminiscent of the categorical semantics of Hofmann’s calculus BC^ω [18] of polytime functions with safe recursion given by the combinator

$$\text{saferec} : \square W \rightarrow W \rightarrow (\square W \rightarrow W \rightarrow W) \rightarrow W$$

on binary strings W , also used in a gluing argument to show soundness w.r.t. polytime functions. In fact, his construction inspired ours for primitive recursion and we expect that the BC^ω can be extended to a dependently typed system using MTT (or MATT) in a similar fashion.

Further variants of BC^ω have been developed (c.f. [17, 19]). However, these critically hinge on linear type systems. Hofmann’s linear recursive calculus for polytime programming has been extended to a dependent type theory (see [2]) using an extension of quantitative type theory (c.f. [3]). It is unclear how such a system might be extended to homotopy types, see, e.g., [28, Sec. 1.7.4] for a discussion for quantitative type theory. Variants of linear homotopy type theory have been developed [28, 30], but there is no general framework such as MTT which admits intensional identity types, linear type formers, dependent types and homotopical interpretations. The closest approximation is [24], but the syntax is complicated and it does not support dependent types yet. Therefore, we do not believe that a univalent type theory for complexity classes using substructural type formers is currently within reach.

An obvious question is whether the Primitive Recursive Dependent Type Theory can be extended to a variant of Homotopy Type Theory (HoTT) [34]. In plain HoTT with univalence added as an axiom there is not much hope since it is not computational. Currently, our best hope is via Cubical Type Theory (CTT) [5, 12]. Its abstract syntax is already formally defined in [33] so it is easy to make the necessary syntactic adjustments to its first universe. However, it is not clear how the soundness proof should be adapted because CTT cannot be interpreted in Set . One could for example require that the structure maps of a cubical set be primitive recursive and embed Set into the resulting category. Even then, retracts of y_N in a category of cubical sets over R remain of homotopy level zero, so $\llbracket U_0 \rrbracket_R$ would require a fundamentally different definition. Another approach would be to define the cubical model in T_{pr} itself.

REFERENCES

- [1] Wilhelm Ackermann. 1928. Zum Hilbertschen Aufbau der reellen Zahlen. *Mathematische Annalen*, 99, 1, (Dec. 1928), 118–133. DOI: [10.1007/BF01459088](https://doi.org/10.1007/BF01459088).
- [2] Robert Atkey. 2023. Polynomial time and dependent types. (2023). arXiv: [2307.09145](https://arxiv.org/abs/2307.09145) [cs.LG].
- [3] Robert Atkey. 2018. Syntax and semantics of quantitative type theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '18)*. Association for Computing Machinery, Oxford, United Kingdom, 56–65. DOI: [10.1145/3209108.3209189](https://doi.org/10.1145/3209108.3209189).
- [4] J. L. Bell. 1989. *The Journal of Symbolic Logic*, 54, 3, 1113–1114. Retrieved Jan. 26, 2024 from <http://www.jstor.org/stable/2274784>.
- [5] Marc Bezem, Thierry Coquand, and Simon Huber. 2014. A model of type theory in cubical sets. In *19th International Conference on Types for Proofs and Programs*. LIPIcs. Leibniz Int. Proc. Inform. Vol. 26. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 107–128. DOI: [10.4230/LIPIcs.TYPES.2013.107](https://doi.org/10.4230/LIPIcs.TYPES.2013.107).
- [6] Ana Bove. 2003. General recursion in type theory. In *Types for Proofs and Programs*. Herman Geuvers and Freek Wiedijk, (Eds.) Springer, Berlin, Heidelberg, 39–58.
- [7] Ana Bove and Venanzio Capretta. 2007. Computation by prophecy. In *International Conference on Typed Lambda Calculus and Applications*.
- [8] Ana Bove and Venanzio Capretta. 2005. Modelling general recursion in type theory. *Mathematical Structures in Computer Science*, 15, 4, 671–708. DOI: [10.1017/S0960129505004822](https://doi.org/10.1017/S0960129505004822).
- [9] Ana Bove and Venanzio Capretta. 2001. Nested general recursion and partiality in type theory. In *Theorem Proving in Higher Order Logics*. Richard J. Boulton and Paul B. Jackson, (Eds.) Springer, Berlin, Heidelberg, 121–125.
- [10] Ana Bove and Venanzio Capretta. 2005. Recursive functions with higher order domains. In *Typed Lambda Calculi and Applications*. Paweł Urzyczyn, (Ed.) Springer, Berlin, Heidelberg, 116–130.
- [11] Venanzio Capretta. 2005. General recursion via coinductive types. *Log. Methods Comput. Sci.*, 1.
- [12] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. 2018. Cubical type theory: a constructive interpretation of the univalence axiom. In *21st International Conference on Types for Proofs and Programs (TYPES 2015)*. LIPIcs. Leibniz Int. Proc. Inform. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern. DOI: [10.4230/LIPIcs.TYPES.2015.5](https://doi.org/10.4230/LIPIcs.TYPES.2015.5).

- [13] Robert L. Constable and Scott F. Smith. 1987. Partial objects in constructive type theory. In *Logic in Computer Science*.
- [14] Rowan Davies and Frank Pfenning. 2001. A modal analysis of staged computation. *J. ACM*, 48, 3, (May 2001), 555–604. DOI: [10.1145/382780.382785](https://doi.org/10.1145/382780.382785).
- [15] Daniel Gratzer, G. A. Kavvos, Andreas Nuyts, and Lars Birkedal. 2021. Multimodal Dependent Type Theory. *Logical Methods in Computer Science*, Volume 17, Issue 3, (July 2021), 7571. DOI: [10.46298/lmcs-17\(3:11\)2021](https://doi.org/10.46298/lmcs-17(3:11)2021).
- [16] Hugo Herbelin and Ludovic Patey. 2014. A calculus of primitive recursive constructions. TYPES abstract. (2014). <https://www.irif.fr/~letouzey/types2014/abstract-41.pdf>.
- [17] Martin Hofmann. 1998. A mixed modal/linear lambda calculus with applications to Bellantoni–Cook safe recursion. In *Computer Science Logic*. Vol. 1414. Mogens Nielsen, Wolfgang Thomas, Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen, (Eds.) Springer, Berlin, Heidelberg, 275–294. DOI: [10.1007/BFb0028020](https://doi.org/10.1007/BFb0028020).
- [18] Martin Hofmann. 1997. An Application of Category-Theoretic Semantics to the Characterisation of Complexity Classes Using Higher-Order Function Algebras. *Bulletin of Symbolic Logic*, 3, 4, (Dec. 1997), 469–486. DOI: [10.2307/421100](https://doi.org/10.2307/421100).
- [19] Martin Hofmann. 2000. Safe recursion with higher types and BCK-algebra. *Annals of Pure and Applied Logic*, 104, 1-3, (July 2000), 113–166. Number: 1-3. DOI: [10.1016/S0168-0072\(00\)00010-5](https://doi.org/10.1016/S0168-0072(00)00010-5).
- [20] Pieter Hofstra and Philip Scott. 2020. Aspects of categorical recursion theory. (Jan. 2020). Retrieved Dec. 1, 2022 from <http://arxiv.org/abs/2001.05778>.
- [21] Stephen Cole Kleene. 1952. *Introduction to metamathematics*. D. Van Nostrand Co., Inc., New York, x+550.
- [22] Nicolai Kraus, Fredrik Nordvall Forsberg, and Chuangjie Xu. 2023. Type-theoretic approaches to ordinals. *Theoretical Computer Science*, 957, 113843. DOI: <https://doi.org/10.1016/j.tcs.2023.113843>.
- [23] Saunders Mac Lane and Ieke Moerdijk. 1994. *Sheaves in Geometry and Logic*. Springer, New York. DOI: [10.1007/978-1-4612-0927-0](https://doi.org/10.1007/978-1-4612-0927-0).
- [24] Daniel R. Licata, Michael Shulman, and Mitchell Riley. 2017. A fibrational framework for substructural and modal logics. In *International Conference on Formal Structures for Computation and Deduction*.
- [25] Maria Maietti. 2010. Joyal’s arithmetic universe as list-arithmetic pretopos. *Theory and Applications of Categories [electronic only]*, 24, (Jan. 2010).
- [26] Lawrence C. Paulson. 1986. Constructing recursion operators in intuitionistic type theory. *Journal of Symbolic Computation*, 2, 4, 325–355. DOI: [10.1016/S0747-7171\(86\)80002-5](https://doi.org/10.1016/S0747-7171(86)80002-5).
- [27] Egbert Rijke. 2022. Introduction to homotopy type theory. (2022). arXiv: [2212.11082](https://arxiv.org/abs/2212.11082) [math.LO].
- [28] Mitchell Riley. 2022. *A Bunched Homotopy Type Theory for Synthetic Stable Homotopy Theory*. PhD Thesis. Wesleyan University.
- [29] Leopoldo Román. 1989. Cartesian categories with natural numbers object. *Journal of Pure and Applied Algebra*, 58, 3, 267–278. DOI: [https://doi.org/10.1016/0022-4049\(89\)90042-X](https://doi.org/10.1016/0022-4049(89)90042-X).
- [30] Urs Schreiber. 2014. Quantization via linear homotopy types. (2014). arXiv: [1402.7041](https://arxiv.org/abs/1402.7041) [math-ph].
- [31] Michael Shulman. 2023. Semantics of multimodal adjoint type theory. (June 2023). Retrieved June 6, 2023 from <http://arxiv.org/abs/2303.02572>.
- [32] Stephen G. Simpson. 2009. *Subsystems of second order arithmetic*. (Second ed.). *Perspectives in Logic*. Cambridge University Press, Cambridge; Association for Symbolic Logic, Poughkeepsie, NY, xvi+444. DOI: [10.1017/CB09780511581007](https://doi.org/10.1017/CB09780511581007).
- [33] Jonathan Sterling. 2021. *First Steps in Synthetic Tait Computability: The Objective Metatheory of Cubical Type Theory*. PhD Thesis. Carnegie Mellon University. DOI: [10.5281/zenodo.6990769](https://doi.org/10.5281/zenodo.6990769). Issue: CMU-CS-21-142.
- [34] The Univalent Foundations Program. 2013. *Homotopy Type Theory: Univalent Foundations of Mathematics*. <https://homotopytypetheory.org/book>, Institute for Advanced Study.

UNIVERSITY OF NOTTINGHAM, UK

Email address: ulrik.buchholtz@nottingham.ac.uk, johannes.schippvonbranitz@nottingham.ac.uk

URL: <https://ulrikbuchholtz.dk/>, <https://jsvb.xyz/>