Realizability at Work: Separating Two Constructive Notions of Finiteness

Marc Bezem
Universitetet i Bergen, Institutt for informatikk, Postboks 7800, N-5020 Bergen, Norway
bezem@ii.uib.no

Thierry Coquand
Chalmers tekniska högskola, Data- och informationsteknik, 412 96 Göteborg, Sweden
coquand@chalmers.se

Keiko Nakata
SAP Innovation Center Network, Konrad-Zuse-Ring 10, 14469 Potsdam, Germany

Erik Parmann
Universitetet i Bergen, Institutt for informatikk, Postboks 7800, N-5020 Bergen, Norway
eparmann@gmail.com

Abstract

We elaborate in detail a realizability model for Martin-Löf dependent type theory with the purpose to analyze a subtle distinction between two constructive notions of finiteness of a set A. The two notions are: (1) A is Noetherian: the empty list can be constructed from lists over A containing duplicates by a certain inductive shortening process; (2) A is streamless: every enumeration of A contains a duplicate.

2012 ACM Subject Classification Theory of computation → Type theory, Theory of computation → Constructive mathematics

Keywords and phrases Type theory, realizability, constructive notions of finiteness

Digital Object Identifier 10.4230/LIPIcs.TYPES.2016.6

Acknowledgements The research for this paper has been carried out for a large part while the first two authors were member of the Institute for Advanced Study in Princeton, as part of the program *Univalent Foundations of Mathematics*. We are grateful for the generous support by the IAS. The paper has benefitted much from comments by Michael Beeson on an early draft.

1 Introduction

We will analyze in detail in type theory the following observation. Let P be a unary predicate of natural numbers. Define

$$\overline{P} = \{n \in \mathsf{N} \mid \forall k < n.\, Pk \vee \neg Pk\}$$

Of course, in classical mathematics $\overline{P} = \mathsf{N}$, but in constructive mathematics this is not true for all P. Let D be a unary predicate of lists over $\overline{P}$ with $D\ell$ expressing that ℓ contains a *duplicate*, that is, two occurrences of the same natural number (which by definition is in $\overline{P}$). Then one can prove constructively that D is inductive in the following sense ("inductive shortening"):

if $D(x :: \ell)$ for all $x \in \overline{P}$, then $D\ell$

The proof is as follows. Let ℓ be a list over $\overline{P}$ and assume $D(x :: \ell)$ for all $x \in \overline{P}$ (IH). We clearly have $D\ell \vee \neg D\ell$, so we can reason by contradiction. Assume $\neg D\ell$. We clearly have $\ell = \mathsf{nil} \vee \neg \ell = \mathsf{nil}$, so we can reason by cases. If $\ell = \mathsf{nil}$, we use $0 \in \overline{P}$ and apply IH to get $D(0 :: \mathsf{nil})$, which is absurd. If $\ell \neq \mathsf{nil}$, then ℓ contains a largest natural number, say $m \in \overline{P}$. If $Pm \vee \neg Pm$, then $m + 1 \in \overline{P}$ and we can apply IH to get $D((m + 1) :: \ell)$, which per construction yields $D\ell$, as $m+1$ is larger than all elements of ℓ, and therefore not duplicating some element of ℓ. This conflicts with the assumption $\neg D\ell$, so we conclude $\neg(Pm \vee \neg Pm)$, which is also absurd, since $\neg\neg(Pm \vee \neg Pm)$ is a constructive tautology. This completes the proof of the inductivity of D.

Since D is inductive in the way above, and $D\mathsf{nil}$ is absurd, D cannot be an inductive bar in the tree of lists over $\overline{P}$. (The concept of an inductive bar making a tree of lists well-founded will be formalized by the concept of a Noetherian relation in Section 3.) This should not come as a surprise, since classically $\overline{P} = \mathsf{N}$, so the tree of lists without duplicates is classically not well-founded, since it is always possible to extend a list without introducing a duplicate. The above argument shows that we can also do this constructively with lists over $\overline{P}$, for any unary predicate P.

In view of the above, only a non-classical axiom can cause $\overline{P}$ to have fewer elements. Of course we cannot downright postulate $\exists n \in \mathsf{N}.\, \neg(Pn \vee \neg Pn)$ without running into inconsistency. But we can consistently postulate $\Phi = \neg\forall n \in \mathsf{N}.\, Pn \vee \neg Pn$. From Φ we immediately infer $\neg\forall n \in \mathsf{N}.\, n \in \overline{P}$, so $\overline{P} \neq \mathsf{N}$. Since one easily (and constructively) sees that $\overline{P}$ is downward closed, Φ "somehow" achieves that $\overline{P}$ is finite. Of course the results in the previous paragraphs still stand, and Φ does not make $\overline{P}$ finite in the sense that D is an inductive bar.

In order to better understand in which way Φ achieves that $\overline{P}$ is finite, assume $f : \mathsf{N} \to \overline{P}$ is an injection. Then we would be able to find arbitrarily large elements in $\overline{P}$, and hence prove $\forall n \in \mathsf{N}.\, Pn \vee \neg Pn$, conflicting with Φ. As a consequence, no $f : \mathsf{N} \to \overline{P}$ is injective. In other words, for every $f : \mathsf{N} \to \overline{P}$ we have $\neg\forall m, n \in \mathsf{N}.\, fm{=}fn \to m{=}n$. By an appeal to Markov's Principle we get $\exists m, n \in \mathsf{N}.\, fm = fn \wedge m \neq n$, that is, there exists a prefix of f which contains a duplicate. If we view lists over $\overline{P}$ not containing duplicates as a tree, then we have just proved that this tree is well-founded, which is another way of saying that $\overline{P}$ is finite. (This notion of finiteness will be made precise by the concept of a streamless relation in Section 3.)

The results in the previous two paragraphs capitalize on Markov's Principle (a weak form of classical reasoning) being consistent with Φ (a non-classical axiom). They are known to co-exist in, for example, the recursive model of type theory. In that model, Φ can be validated by the unsolvability of the halting problem. Although this model has been known for quite some time, it is an important side-goal of this paper to give a detailed account. Our main objective is to formalize the argument above in type theory, and prove that finiteness based on equality being Noetherian is strictly stronger than finiteness based on equality being streamless. This confirms a conjecture formulated by Coquand and Spiwack in [3]. We also give a novel proof that every Noetherian relation is streamless. This proof is due to the last author [11, Chapter 4] and formalized in Coq [10].

In type theory, a subset of N is a type $\Sigma x{:}\mathsf{N}.\, Px$ given a type family $P : \mathsf{N} \to \mathsf{U}$. Elements of this Σ-type (to be defined in Section 2) are pairs (n, p) consisting of a natural number n and a proof $p : Pn$. It may happen that also $p' : Pn$, with p' different from p. This phenomenon is called *proof relevance*. We do not want to count (n, p) and (n, p') as two elements of the subset of N defined by P. Therefore we only count the first projections of objects in $\Sigma x{:}\mathsf{N}.\, Px$. Another approach would be to take the type $\Sigma x{:}\mathsf{N}.\, \|Px\|$, where $\|\_\|$ stands for propositional truncation, a way of making all inhabitants of Px indistinguishable, see [15, Section 3.7].

The remainder of the paper is organized as follows. In Section 2, we define the basic type theory. We introduce Noetherian relations and streamless relations in Section 3 and prove that any Noetherian relation is streamless. The realizability model is constructed in Section 4, with realizers for Markov's Principle in Section 5 and for the unsolvability of the halting problem in Section 6. This shows that the type theory can be consistently extended with these two axioms. Then, in Section 7 we show that it cannot be proved in type theory that any streamless set is Noetherian. We conclude with a discussion of related work, in particular the Kleene Tree, in Section 8. For readers already familiar with dependent type theory and inductive bars it might be efficient to read the conclusion first.

2 Dependent Type Theory

We closely follow the approach of Coquand and Spiwack [3]. We define Martin-Löf dependent type theory as a set of typing rules defining a typing relation $\Gamma \vdash M : A$ where M and A are terms in an extension Λ of the untyped lambda calculus, and Γ is a context. A *context* is a sequence $x_1 : A_1, \dots, x_n : A_n$, where the x_i are pairwise distinct variables and the A_i are terms of Λ (representing types). The approach of [3] makes the construction of a realizability model easier: all typable terms are already terms of Λ realizing their types, and their computational behaviour can be studied in Λ.

An important aspect is that the type theory is open-ended, new constants and inductive definitions can be (and will be) added. If the type theory is extended, then also Λ is extended, and the realizability model is extended accordingly. We start by describing the main characteristics of Λ.

2.1 The Underlying Computational System

The calculus Λ is an extension of the untyped lambda calculus with two sorts of constants, *constructors* and *operators*. Constructors typically represent types (e.g., the type of the natural numbers), type forming constructions (e.g., the sum of two types), and term forming constructions (e.g., $\mathsf{0}$, S, nil). Operators typically represent destructors (e.g., recursors), operations (e.g., the length of a list), and convenient abbreviations.

The abstract syntax for terms of Λ is

$$M, N ::= x \mid \lambda x.\, M \mid M\, N \mid c \mid o,$$

where c is the syntactic category of the constructors and o that of the operators. We write $\mathsf{FV}(M)$ to denote the set of free variables in M; we call M closed if $\mathsf{FV}(M)$ is empty. The computational behavior of the terms is determined by β-reduction plus so-called ι-reduction rules. The latter are left-linear and mutually disjoint (non-overlapping), ensuring confluence of $\beta\iota$-reduction [8]. (Confluence is important to warrant the correct interpretation of elements of an inductive type as $\beta\iota$-equivalence classes of terms. For example, the normal forms $\mathsf{0}$ and $\mathsf{S0}$ are in different classes because of confluence.) All ι-reduction rules are of the form

$$o\, p_1 \dots p_k = M,$$

where o is an operator and $p_1, \dots, p_k$ are so-called constructor patterns. For any ι-reduction rule we require $\mathsf{FV}(M) \subseteq \mathsf{FV}(o\, p_1 \dots p_k)$, that is, no new variables can be introduced. Constructor patterns $p_1, \dots, p_k$ are defined by the following abstract syntax

$$p ::= x \mid c\, p_1 \dots p_l,$$

where c is a constructor.

$$\frac{}{\vdash} \qquad \frac{\Gamma \vdash A}{\Gamma, x : A \vdash}$$

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathsf{U}} \qquad \frac{\Gamma \vdash A : \mathsf{U}}{\Gamma \vdash A} \qquad \frac{\Gamma \vdash A \quad \Gamma, x : A \vdash B}{\Gamma \vdash \Pi x{:}A.\, B}$$

$$\frac{\Gamma \vdash}{\Gamma \vdash x : A}\ x{:}A \in \Gamma \qquad \frac{\Gamma \vdash M : A \quad \Gamma \vdash B}{\Gamma \vdash M : B}\ A =_{\beta\iota} B$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash \Pi x{:}A.\, B \quad \Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.\, M : \Pi x{:}A.\, B} \qquad \frac{\Gamma \vdash M : \Pi x{:}A.\, B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B(N)}$$

$$\frac{\Gamma \vdash A : \mathsf{U} \quad \Gamma, x : A \vdash B : \mathsf{U}}{\Gamma \vdash \Pi x{:}A.\, B : \mathsf{U}}$$

Figure 1 General typing rules of Martin-Löf type theory with universe U.

Constructors, as well as operators with their ι-reduction rules, will be introduced in the sequel, as need arises, always complying with the above syntax.

Notational conventions.

We tacitly assume capture-free substitution and consider terms up to α-conversion. We write $M =_{\beta\iota} N$ or just $M = N$ if M and N are $\beta\iota$-convertible. By $M(x/N)$ we denote the result of substituting N for all the free occurences of the variable x in M. We may write $M(N)$ if the variable x is clear from the context. For example, $(\lambda x.\, M)N = M(x/N)$ and $(\lambda x.\, M)N = M(N)$ both denote a β-step. We abbreviate $(\lambda x.\, xx)(\lambda x.\, xx)$ to Ω.

2.2 General rules of the type theory

There are three forms of judgments in the type theory:

$$\Gamma \vdash \quad \text{and} \quad \Gamma \vdash A \quad \text{and} \quad \Gamma \vdash M : A.$$

The judgment $\Gamma \vdash$ means that Γ is a well-typed context, $\Gamma \vdash A$ means that the type A is well-formed in the context Γ, and $\Gamma \vdash M : A$ means that the term M has the type A in the context Γ. We (mostly) use metavariables A, B for types, and M, N for terms, but recall that they are all terms of Λ.

For the general rules, we have a constructor U for the universe since we want type families to be first-class citizens. We add an operator Pi for dependent products, with ι-reduction $\mathsf{Pi}\, A\, B\, x = B\, x$. For readability, we write $\Pi x{:}A.\, B$ instead of $\mathsf{Pi}\, A\, (\lambda x.\, B)$, and $A \to B$ instead of $\mathsf{Pi}\, A\, (\lambda x.\, B)$ if x does not occur free in B. The typing rules are the standard rules for the Martin-Löf type theory, given in Figure 1.

We can derive, for example, $A : \mathsf{U} \vdash$, and so $\vdash \mathsf{U} \to \mathsf{U}$, and in some more steps $A : \mathsf{U} \vdash A \to \mathsf{U}$. The latter $A \to \mathsf{U}$ is the type of unary predicates on a type A (also called *type families* over A). The former $\mathsf{U} \to \mathsf{U}$ is the type of functions on the universe. Both are *large* types, i.e., types not in U. Types in U are called *small* types.

2.3 Specific rules of the type theory

We extend the type theory by specific inductive types, which are all standard. We add constants and give typing rules, as well as ι-reduction rules for the operators.

Empty type

We define the empty type with no constructors:

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathsf{N}_0 : \mathsf{U}}$$

and its elimination rule (also known as the *ex falso* rule):

$$\frac{\Gamma \vdash A : \mathsf{U}}{\Gamma \vdash \mathsf{ExF} : \mathsf{N}_0 \to A}$$

We define negation as the abbreviation $\neg := \lambda A.\, A \to \mathsf{N}_0$.

Sum

We have the sum type with its two constructors:

$$\frac{\Gamma \vdash A : \mathsf{U} \quad \Gamma \vdash B : \mathsf{U}}{\Gamma \vdash A + B : \mathsf{U}} \qquad \frac{\Gamma \vdash A : \mathsf{U} \quad \Gamma \vdash A + B : \mathsf{U}}{\Gamma \vdash \mathsf{inl} : A \to A + B} \qquad \frac{\Gamma \vdash B : \mathsf{U} \quad \Gamma \vdash A + B : \mathsf{U}}{\Gamma \vdash \mathsf{inr} : B \to A + B}$$

and may perform case analysis on terms of type $A + B$:

$$\frac{\Gamma \vdash A : \mathsf{U} \quad \Gamma \vdash B : \mathsf{U} \quad \Gamma \vdash C : \mathsf{U} \quad \Gamma \vdash (A + B) : \mathsf{U}}{\Gamma \vdash \mathsf{case} : (A \to C) \to (B \to C) \to A + B \to C}$$

where the ι-reductions are given by:

$$\begin{aligned} \mathsf{case}\, M\, N\, (\mathsf{inl}\, a) &= M\, a \\ \mathsf{case}\, M\, N\, (\mathsf{inr}\, b) &= N\, b \end{aligned}$$

With negation and sum type used for constructive disjunction we can define the concept of decidability that will play an important role in the sequel.

▶ **Definition 1.** We call a type $A : \mathsf{U}$ *decidable* if $A : \mathsf{U} \vdash M : A + \neg A$ for some M. The type $A + \neg A$ will often be abbreviated by $\mathsf{dec}\, A$. In such cases we also say that $\mathsf{dec}\, A$ is *inhabited*, without explicit reference to Γ or M. Predicates are called decidable if they are pointwise decidable. For example, $P : A \to \mathsf{U}$ is decidable if $\Pi a{:}A.\, \mathsf{dec}\,(Pa)$ is inhabited; $R : A \to A \to \mathsf{U}$ is decidable if $\Pi a, a'{:}\, A.\, \mathsf{dec}\,(Raa')$ is inhabited. The latter is an example of how we denote two Π-abstractions with the same base type A.

Unit type

We have the unit type with one single constructor:

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathsf{N}_1 : \mathsf{U}} \qquad \frac{\Gamma \vdash}{\Gamma \vdash 0 : \mathsf{N}_1}$$

Booleans

We have the type for Booleans with two constructors:

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathsf{N}_2 : \mathsf{U}} \qquad \frac{\Gamma \vdash}{\Gamma \vdash \mathsf{0} : \mathsf{N}_2} \qquad \frac{\Gamma \vdash}{\Gamma \vdash \mathsf{1} : \mathsf{N}_2}$$

and a conditional expression:

$$\frac{\Gamma \vdash C : \mathsf{N}_2 \to \mathsf{U}}{\Gamma \vdash \mathsf{brec} : C\,\mathsf{0} \to C\,\mathsf{1} \to \Pi b{:}\mathsf{N}_2.\, C\, b}$$

with the ι-reduction given by

$$\begin{array}{lcl} \mathsf{brec}\, M\, N\, \mathsf{0} & = & M \\ \mathsf{brec}\, M\, N\, \mathsf{1} & = & N \end{array}$$

Note that brec does not make it possible to define a function $f : \mathsf{N}_2 \to \mathsf{U}$ with, e.g., $f\, i = \mathsf{N}_i$, since that would require $C = \lambda b.\, \mathsf{U}$, which cannot be typed. Therefore, certain useful operators have to be defined ad-hoc. Here we define a decidable equality for Booleans:

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathsf{beq} : \mathsf{N}_2 \to \mathsf{N}_2 \to \mathsf{U}}$$

whose ι-reductions are given by:

$$\begin{array}{llllll} \mathsf{beq} & \mathsf{0} & \mathsf{0} & = & \mathsf{N}_1 \\ \mathsf{beq} & \mathsf{0} & \mathsf{1} & = & \mathsf{N}_0 \\ \mathsf{beq} & \mathsf{1} & \mathsf{0} & = & \mathsf{N}_0 \\ \mathsf{beq} & \mathsf{1} & \mathsf{1} & = & \mathsf{N}_1 \end{array}$$

Natural numbers

We have the type N with the two well-known constructors:

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathsf{N} : \mathsf{U}} \qquad \frac{\Gamma \vdash}{\Gamma \vdash \mathsf{0} : \mathsf{N}} \qquad \frac{\Gamma \vdash}{\Gamma \vdash \mathsf{S} : \mathsf{N} \to \mathsf{N}}$$

Notice that $\mathsf{0}$ is ad-hoc polymorphic and is a constructor of $\mathsf{N}_1, \mathsf{N}_2$ and N. We have the recursor (dependent eliminator) rec:

$$\frac{\Gamma \vdash C : \mathsf{N} \to \mathsf{U}}{\Gamma \vdash \mathsf{rec} : C\,\mathsf{0} \to (\Pi n{:}\mathsf{N}.\, C\, n \to C\,(S\, n)) \to \Pi n{:}\mathsf{N}.\, C\, n}$$

with ι-reductions given by

$$\begin{array}{lcl} \mathsf{rec}\, M\, N\, \mathsf{0} & = & M \\ \mathsf{rec}\, M\, N\, (\mathsf{S}\, n) & = & N\, n\, (\mathsf{rec}\, M\, N\, n). \end{array}$$

We also define a decidable equality on natural numbers:

$$\frac{\Gamma \vdash}{\Gamma \vdash \mathsf{eq} : \mathsf{N} \to \mathsf{N} \to \mathsf{U}}$$

whose ι-reductions are given by:

$$\begin{array}{lcccl} \mathsf{eq} & \mathsf{0} & \mathsf{0} & = & \mathsf{N}_1 \\ \mathsf{eq} & (\mathsf{S}\, x) & \mathsf{0} & = & \mathsf{N}_0 \\ \mathsf{eq} & \mathsf{0} & (\mathsf{S}\, x) & = & \mathsf{N}_0 \\ \mathsf{eq} & (\mathsf{S}\, x) & (\mathsf{S}\, y) & = & \mathsf{eq}\ x\ y. \end{array}$$

By double induction one can easily prove:

▶ **Lemma 2.** *There exist proofs $deq_{\mathsf{N}_2}, deq_{\mathsf{N}}$ such that*

$$\vdash deq_{\mathsf{N}_2} : \Pi x, y{:}\,\mathsf{N}_2.\,\mathsf{dec}\,(\mathsf{beq}\,x\,y) \qquad\qquad \vdash deq_{\mathsf{N}} : \Pi n, m{:}\,\mathsf{N}.\,\mathsf{dec}\,(\mathsf{eq}\,n\,m)$$

Lists

We have the usual type of lists over A, denoted by $[A]$:

$$\frac{\Gamma \vdash A : \mathsf{U}}{\Gamma \vdash [A] : \mathsf{U}} \qquad \frac{\Gamma \vdash A : \mathsf{U}}{\Gamma \vdash \mathsf{nil} : [A]} \qquad \frac{\Gamma \vdash A : \mathsf{U}}{\Gamma \vdash \mathsf{cons} : A \to [A] \to [A]}$$

and the list recursor, writing $a :: l$ for $\mathsf{cons}\,a\,l$ here and below:

$$\frac{\Gamma \vdash A \quad \Gamma \vdash C : [A] \to \mathsf{U}}{\Gamma \vdash \mathsf{lrec} : C\,\mathsf{nil} \to (\Pi a{:}A.\,\Pi l{:}[A].\,C\,l \to C(a :: l)) \to \Pi l{:}[A].\,C\,l}$$

with ι-reductions given by

$$\begin{array}{lcl} \mathsf{lrec}\,M\,N\,\mathsf{nil} & = & M \\ \mathsf{lrec}\,M\,N\,(a :: l) & = & N\,a\,l\,(\mathsf{lrec}\,M\,N\,l). \end{array}$$

Dependent pairs

We have the Σ-type for dependent pairs:

$$\frac{\Gamma \vdash A : \mathsf{U} \quad \Gamma, x : A \vdash B : \mathsf{U}}{\Gamma \vdash \Sigma x{:}A.\,B : \mathsf{U}} \qquad \frac{\Gamma \vdash \Sigma x{:}A.\,B : \mathsf{U} \quad \Gamma \vdash W : A \quad \Gamma \vdash P : B(W)}{\Gamma \vdash (W, P) : \Sigma x{:}A.\,B}$$

with dependent eliminator (recursor):

$$\frac{\Gamma \vdash A : \mathsf{U} \quad \Gamma, x : A \vdash B : U \quad \Gamma \vdash \Sigma x{:}A.\,B : \mathsf{U} \quad \Gamma \vdash C : (\Sigma x{:}A.\,B) \to \mathsf{U}}{\Gamma \vdash \mathsf{srec} : (\Pi x{:}A.\,\Pi p{:}B.\,C\,(x,p)) \to \Pi y{:}(\Sigma x{:}A.\,B).\,C\,y}$$

with ι-reduction given by:

$$\mathsf{srec}\,Q\,(w,p) \quad = \quad Q\,w\,p$$

We use similar notational conventions for $\Sigma x{:}A.\,B$ as for $\Pi x{:}A.\,B$. This means that the actual syntax is $\mathsf{Sig}\,A\,(\lambda x.\,B)$. When x does not appear free in B, we may abbreviate $\Sigma x{:}A.\,B$ as $A \times B$.

3 Noetherian relations and streamless relations

In this section we define the concepts Noetherian and streamless for relations. When applied to the equality relation on a type A, they yield two classically equivalent definitions of finiteness. We first extend the type theory with new constants and rules to facilitate these definitions.

3.1 Auxiliary constants and rules

Given a list l of elements of a type A and a predicate P on A, we define a predicate $\mathsf{exists}\,P\,l$ to be true if l contains an element that satisfies P. Formally, we add a typing rule for exists

$$\frac{\Gamma \vdash A : \mathsf{U}}{\Gamma \vdash \mathsf{exists} : (A \to \mathsf{U}) \to [A] \to \mathsf{U}}$$

and ι-reductions given by:

$$\begin{array}{lcl} \mathsf{exists}\, P\, \mathsf{nil} & = & \mathsf{N}_0 \\ \mathsf{exists}\, P\, (a :: l) & = & (P\, a) + \mathsf{exists}\, P\, l \end{array}$$

Note that exists is not definable by list recursion since it would require $C = \lambda l.\, \mathsf{U}$, which cannot be typed. A similar remark holds for good which we define now.

Given a binary relation R on a type A and a list l over A, we define a predicate $\mathsf{good}\, R\, l$ to be true if l contains elements that are related by R in the same order in which they occur in l. Formally, we define a typing rule for good by

$$\frac{\Gamma \vdash A : \mathsf{U}}{\Gamma \vdash \mathsf{good} : (A \to A \to \mathsf{U}) \to [A] \to \mathsf{U}}$$

and ι-reductions by:

$$\begin{array}{lcl} \mathsf{good}\, R\, \mathsf{nil} & = & \mathsf{N}_0 \\ \mathsf{good}\, R\, (a :: l) & = & \mathsf{exists}\, (R\, a)\, l + \mathsf{good}\, R\, l \end{array}$$

The following functions are actually definable by the recursors in Section 2.3. We define the length function on lists:

$$\frac{\Gamma \vdash A : \mathsf{U}}{\Gamma \vdash \mathsf{length} : [A] \to \mathsf{N}}$$

with ι-reductions given by:

$$\begin{array}{r} \mathsf{length}\, \mathsf{nil} = 0 \\ \mathsf{length}\, (h :: t) = \mathsf{S}(\mathsf{length}\, t) \end{array}$$

Given a function f from natural numbers to a type A, the function $\mathsf{take}\, f$ returns for every natural number n the list consisting of the first n values of f. Formally, we define a typing rule for take by

$$\frac{\Gamma \vdash A : \mathsf{U}}{\Gamma \vdash \mathsf{take}\ : (\mathsf{N} \to A) \to \mathsf{N} \to [A]}$$

and, writing $\overline{f}n$ for $(\mathsf{take}\, f\, n)$, ι-reductions:

$$\begin{array}{lcl} \overline{f}0 & = & \mathsf{nil} \\ \overline{f}(S\, n) & = & (f\, n) :: (\overline{f}n). \end{array}$$

Given a type A and a predicate P on $[A]$, we define by induction the predicate $\mathsf{bar}\, A\, P$ of lists over A that are "barred" by P. The classical intuition is that a list is "barred" by P if every extension eventually satisfies P. More precisely, by the inductive definition below, $\mathsf{bar}\, A\, P$ is the smallest predicate which contains P and which is closed under inductive shortening, that is, holds of l whenever it holds of $a :: l$ for all $a : A$. Since the type A will be the same in this section, we will use the abbreviation bAr for $\mathsf{bar}\, A$, where the capital in bAr should remind the reader of the implicit argument A. We add the following typing rules for proving that the list l is barred by P:

$$\frac{\Gamma \vdash A : \mathsf{U}}{\Gamma \vdash \mathsf{bAr} : ([A] \to \mathsf{U}) \to [A] \to \mathsf{U}}$$

$$\frac{\Gamma \vdash A : \mathsf{U} \quad \Gamma \vdash l : [A] \quad \Gamma \vdash P : [A] \to \mathsf{U} \quad \Gamma \vdash X : P\, l}{\Gamma \vdash \mathsf{base}\, X : \mathsf{bAr}\, P\, l}$$

$$\frac{\Gamma \vdash A : \mathsf{U} \quad \Gamma \vdash l : [A] \quad \Gamma \vdash P : [A] \to \mathsf{U} \quad \Gamma \vdash Y : \Pi a{:}A.\, \mathsf{bAr}\, P\, (a :: l)}{\Gamma \vdash \mathsf{step}\, Y : \mathsf{bAr}\, P\, l}$$

The eliminator for $\mathsf{bAr}\,P$ will be called a bar recursor, but should not be confused with Spector's bar recursor from [13]. The latter is an ingenious combinator of much greater proof theoretic strength than the one here. Our bar recursor has the following type:

$$\frac{\Gamma \vdash A : U \quad \Gamma \vdash P : [A] \to U \quad \Gamma \vdash C : [A] \to U}{\Gamma \vdash \mathsf{barrec} : (\Pi l{:}[A].\, P\,l \to C\,l) \to (\Pi l{:}[A].\,(\Pi a{:}A.\, C\,(a :: l)) \to C\,l) \to \Pi l{:}[A].\,\mathsf{bAr} P\,l \to C\,l}$$

and its computational behaviour is described by the following ι-reductions:

$$\begin{array}{lcl} \mathsf{barrec}\,B\,S\,l\,(\mathsf{base}\,X) & = & B\,l\,X \\ \mathsf{barrec}\,B\,S\,l\,(\mathsf{step}\,Y) & = & S\,l\,(\lambda a.\,\mathsf{barrec}\,B\,S\,(a :: l)\,(Y\,a)). \end{array}$$

3.2 Definition of streamless and Noetherian

Streamless and Noetherian can both be defined as properties of a binary relation $R : A \to A \to \mathsf{U}$. Informally speaking, R is streamless if every stream, i.e., infinite sequence in A, contains two elements related by R in the reversed order as they appear in the sequence.[1]

▶ **Definition 3.** A relation $R : A \to A \to \mathsf{U}$ on a type $A : \mathsf{U}$ is *streamless* if every function $f : \mathsf{N} \to A$ from natural numbers to A has a prefix that is R-good, i.e.,

$$\mathsf{streamless}\,R := \Pi f{:}\mathsf{N} \to A.\,\Sigma n{:}\mathsf{N}.\,\mathsf{good}\,R\,(\overline{f}n)$$

The equality relation on A being streamless expresses that every stream contains a duplicate, i.e., is constructively non-injective. This is classically equivalent to saying that A is finite.

Noetherian is not so easily explained, it is an acquired taste. We call P an *inductive bar* in $[A]$ if $\mathsf{bAr}\,P\,\mathsf{nil}$ holds, that is, if nil is barred by P. By the inductive definition of $\mathsf{bAr}\,P$ this means that nil can be obtained from lists satisfying P by inductive shortening. We call a relation R Noetherian if $(\mathsf{good}\,R)$ is an inductive bar.

▶ **Definition 4.** A relation $R : A \to A \to \mathsf{U}$ on a type $A : \mathsf{U}$ is *Noetherian* if $\mathsf{bAr}\,(\mathsf{good}\,R)\,\mathsf{nil}$ holds:

$$\mathsf{Noetherian}\,R := \mathsf{bAr}\,(\mathsf{good}\,R)\,\mathsf{nil}$$

The equality relation on A being Noetherian is also classically equivalent to A being finite. However, unlike streamless, Noetherian allows us to use induction on bAr. It is exactly this which allows us to prove that Noetherian relations are streamless. The novelty of this proof is that it does not use a relation $l \prec f$ of a list being a prefix of a function, and hence not an equality relation on the type A. Of course, adding equality would not be problematic, but is it somehow pleasing that equality is not used for proving a result that does not involve equality (the normal form of any such proof would not involve equality anyway). A nice corollary of the next theorem is that every Noetherian relation is reflexive, a fact that would otherwise require a non-trivial proof.

▶ **Theorem 5.** *There is a proof M such that*

$$A : \mathsf{U}, R : A \to A \to \mathsf{U} \vdash M : \mathsf{Noetherian}\,R \to \mathsf{streamless}\,R$$

[1] The name *streamless* may well be considered a misnomer if R is not an equality relation. Classically, streamless means that there is no f such that $i > j \to \overline{R}ij$ for all i, j ($\overline{R}$ the complement of R).

Proof. We prove in the context $A : \mathsf{U}, R : A \to A \to \mathsf{U}, f : \mathsf{N} \to A$ that the following type is inhabited:

$$\Pi l{:}[A].\, \mathsf{bAr}\,(\mathsf{good}\,R)\,l \to \mathsf{subG}\,R\,f\,l \to \mathsf{subEx}\,R\,f\,l \to \Sigma m{:}\mathsf{N}.\,\mathsf{good}\,R\,(\overline{f}m), \tag{1}$$

where $\mathsf{subG}\,R\,f\,l$ and $\mathsf{subEx}\,R\,f\,l$ are defined as the following abbreviations.

$$\begin{aligned} \mathsf{subG}\,R\,f\,l &:= \mathsf{good}\,R\,l \to \mathsf{good}\,R\,(\overline{f}(\mathsf{length}\,l)) \\ \mathsf{subEx}\,R\,f\,l &:= \Pi a{:}A.\,(\mathsf{exists}\,(R\,a)\,l \to \mathsf{exists}\,(R\,a)\,(\overline{f}(\mathsf{length}\,l))) \end{aligned}$$

These abbreviations express that the $\mathsf{good}/\mathsf{exists}$ properties of l imply those of $\overline{f}(\mathsf{length}\,l)$, which will suffice to show (1) . Also note that they both trivially hold for nil, so that (1) with $l = \mathsf{nil}$ implies Theorem 5. (Note that both are trivially implied by $l = \overline{f}(\mathsf{length}\,l)$.)

We prove (1) by induction on $\mathsf{bAr}\,(\mathsf{good}\,R)\,l$, that is, using the last rule of the previous section with the predicate $P := (\mathsf{good}\,R)$ and the predicate

$$C := \lambda l{:}[A].\,\mathsf{subG}\,R\,f\,l \to \mathsf{subEx}\,R\,f\,l \to \Sigma m{:}\mathsf{N}.\,\mathsf{good}\,R\,(\overline{f}m).$$

Thus the proof of (1) will be of the form $\mathsf{barrec}\,H_b\,H_s$ with $H_b : \Pi l{:}[A].\,P\,l \to C\,l$ and

$$H_s : \Pi l{:}[A].\,(\Pi a{:}A.\,C(a :: l)) \to Cl,$$

corresponding to the base case and the step case, respectively, elaborated below.

Base case. To construct $H_b : \Pi l{:}[A].\,P\,l \to C\,l$, assume $l : [A]$ such that $\mathsf{good}\,R\,l$, $\mathsf{subG}\,R\,f\,l$, and $\mathsf{subEx}\,R\,f\,l$. From $\mathsf{subG}\,R\,f\,l$ and $\mathsf{good}\,R\,l$ we immediately get the goal $\mathsf{good}\,R\,(\overline{f}(\mathsf{length}\,l))$.

Step case. To construct $H_s : \Pi l{:}[A].\,(\Pi a{:}A.\,C(a :: l)) \to Cl$, assume $l : [A]$ such that $\Pi a{:}A.\,C(a :: l)$. We have to show $C\,l$. The latter expands to $\mathsf{subG}\,R\,f\,l \to \mathsf{subEx}\,R\,f\,l \to \Sigma m{:}\mathsf{N}.\,\mathsf{good}\,R\,(\overline{f}m)$, so we assume $\mathsf{subG}\,R\,f\,l$ and $\mathsf{subEx}\,R\,f\,l$, and show $\Sigma m{:}\mathsf{N}.\,\mathsf{good}\,R\,(\overline{f}m)$. Expanding the induction hypothesis $(\Pi a{:}A.\,C(a :: l))$ yields

$$\Pi a{:}A.\,\mathsf{subG}\,R\,f\,(a :: l) \to \mathsf{subEx}\,R\,f\,(a :: l) \to \Sigma m{:}\mathsf{N}.\,\mathsf{good}\,R\,(\overline{f}m).$$

We apply this to $a = f(\mathsf{length}\,l)$, and proceed to proving the two assumptions in the following two subcases.

Subcase $\mathsf{subG}\,R\,f\,(f(\mathsf{length}\,l) :: l)$. Expanding the abbreviation subG we get

$$\mathsf{good}\,R\,(f(\mathsf{length}\,l) :: l) \to \mathsf{good}\,R\,(\overline{f}(\mathsf{length}\,(f(\mathsf{length}\,l) :: l))).$$

Since $(\mathsf{length}\,(f(\mathsf{length}\,l) :: l))$ reduces to $\mathsf{S}(\mathsf{length}\,l)$, the conclusion of the above formula reduces to $\mathsf{good}\,R\,(f(\mathsf{length}\,l) :: \overline{f}(\mathsf{length}\,l))$. Using the definition of good in both the antecedent and the consequent it becomes clear that we have to prove:

$$\begin{aligned} &(\mathsf{exists}\,(R\,(f\,(\mathsf{length}\,l)))\,l + \mathsf{good}\,R\,l) \to \\ &(\mathsf{exists}\,(R\,(f(\mathsf{length}\,l)))\,(\overline{f}(\mathsf{length}\,l)) + \mathsf{good}\,R\,(\overline{f}(\mathsf{length}\,l))). \end{aligned}$$

The latter is easily proved by cases using the assumption $\mathsf{subEx}\,R\,f\,l$ with $a = f(\mathsf{length}\,l)$ for the left summand and the assumption $\mathsf{subG}\,R\,f\,l$ for the right summand.

Subcase $\mathsf{subEx}\,R\,f\,(f(\mathsf{length}\,l) :: l)$. Expanding the abbreviation subEx and reducing we get

$$\begin{aligned} &\Pi a{:}A.\ \mathsf{exists}\,(R\,a)\,(f(\mathsf{length}\,l) :: l) \to \\ &\qquad\qquad \mathsf{exists}\,(R\,a)\,(f(\mathsf{length}\,l) :: \overline{f}(\mathsf{length}\,l)). \end{aligned}$$

Using the definition of exists we get by reducing

$$\Pi a{:}A.\ ((R\,a\,(f(\mathsf{length}\,l))) + \mathsf{exists}\,(R\,a)\,l) \to \\ ((R\,a\,(f(\mathsf{length}\,l))) + \mathsf{exists}\,(R\,a)\,(\overline{f}(\mathsf{length}\,l))).$$

The latter follows easily from the assumption $\mathsf{subEx}\,R\,f\,l$. This finishes the second subcase of the step case and we are done. ◀

4 The Model Construction

In this section, we construct the realizability model for the type theory, based on the underlying computational system Λ. Terms are interpreted by $\beta\iota$-equivalence classes of the terms of Λ. Types are interpreted by sets of such equivalence classes. Typically, if $\Gamma \vdash M : A$, then the interpretation of M is an element of the interpretation of A (both relative to the interpretation of Γ).

We use the realizability model to show the unprovability of the converse of Theorem 5. For this result, we use (the functional version of) Markov's principle and a non-classical axiom $\neg\Pi n{:}\mathsf{N}.\,(H n) + \neg(H n)$ for a halting predicate H. Both will be shown to be true in the model in later sections.

4.1 Pointed DCPOs and fixpoints

We recall Pataraia's fixpoint theorem [12], which states that every monotone endofunction[2] on a pointed directed complete partial order (DCPO) has a least fixpoint.

▶ **Definition 6.** Let $(P, \leq)$ be a partial order. A subset X of P is *directed* if it is nonempty and, for every $x, y \in X$, there exists $z \in X$ such that $x \leq z$ and $y \leq z$.

▶ **Definition 7.** A partial order $(P, \leq)$ is a *directed complete partial order* (*DCPO*) if every directed subset of P has a least upper bound (supremum) in P. A DCPO $(P, \leq)$ is *pointed* if the empty set has a supremum, which is then the least element $\bot_P$ of P.

▶ **Definition 8.** An endofunction $f : P \to P$ is *monotone* if it is order-preserving, i.e., for every $x, y \in P$, $x \leq y$ implies $f(x) \leq f(y)$.

▶ **Theorem 9.** *Every monotone endofunction function on a pointed DCPO has a least fixpoint, which is also the least pre-fixpoint.*

A short proof can be found in [6]. The standard argument, transfinite iteration of the function starting at the least element, also works. The reason is that the transfinite sequence is directed.

We will use Pataraia's Theorem with the following DCPO. Let D be a set. Elements of the DCPO are pairs (S, F) where $S \in \mathcal{P}(D)$, that is, S is a subset of D, and F is a function $S \to \mathcal{P}(D)$, viewed as a single-valued set of pairs. We can order such pairs by $(S, F) \leq (S', F')$ if $S \subseteq S'$ and $F \subseteq F'$. The latter conjunct can be rephrased by saying that F is the restriction of F' to S. This does *not* yield a complete lattice on pairs (S, F), even though $(\mathcal{P}(D), \subseteq)$ is. For example, if $D = \{d, e\}$ and $S = \{d\}$ and F_d maps d to $\{d\}$ and F_e maps d to $\{e\}$, then there is no F such that $F_d, F_e \subseteq F$. (It is tempting but wrong to think that F mapping d to $\{d, e\}$ extends F_d and F_e.)

[2] An *endofunction* is a function with the same domain and codomain.

However, if X is a directed set of pairs (S, F), then the pair

$$(\bigcup_{(S,F)\in X} S\,,\ \bigcup_{(S,F)\in X} F)$$

is the least upper bound of X. Note that, since X is directed, $\bigcup_{(S,F)\in X} F$ is a function from $\bigcup_{(S,F)\in X} S$ to $\mathcal{P}(D)$ as required. If $X = \emptyset$, we get the least element $(\emptyset, \emptyset)$ by the same formula above. It follows that the set of pairs ordered as above is a pointed DCPO. Consequently, every monotone endofunction has a least fixpoint.

4.2 Realizability model

We shall now define the realizability model. The *domain* D is the set of terms in the extended untyped lambda calculus modulo $\beta\iota$-equality. Hence, elements of D are equivalence classes of terms. For simplicity, however, we will often call them just terms, and write M to denote the equivalence class of M. Next, we define which elements of D represent types and how to find the subset of elements associated with each type.

We shall first prepare the interpretation of the inductive types. Define $\mathsf{Num} \subseteq \mathsf{D}$ as the smallest set containing $\mathsf{0}$ and closed under the successor, i.e., $\mathsf{S}\,n$ is in Num if n is in Num. Formally, we define Num as the least fixpoint of the following monotone endofunction Ψ_{Num} on $\mathcal{P}(\mathsf{D})$:

$$\Psi_{\mathsf{Num}}(X) := \{\mathsf{0}\} \cup \{\mathsf{S}\,n \mid n \in X\}.$$

The poset $(\mathcal{P}(\mathsf{D}), \subseteq)$ is a complete lattice, hence the least fixpoint exists by the Knaster-Tarski theorem, which is a special case of Pataraia's theorem.

Similarly, given a set $A \subseteq \mathsf{D}$, we define $\mathsf{List}(A) \subseteq \mathsf{D}$ as the least fixpoint of the following monotone endofunction $\Psi_{\mathsf{List}(A)}$ on $(\mathcal{P}(\mathsf{D}) \subseteq)$:

$$\Psi_{\mathsf{List}(A)}(X) := \{\mathsf{nil}\} \cup \{\mathsf{cons}\,a\,l \mid a \in A \wedge l \in X\}.$$

Informally, $\mathsf{List}(A)$ consists of classes that are $\beta\iota$-equivalent to lists over A.

Note that bAr is an inductively defined function on $[A] \to \mathsf{U}$. Given a set $A \subseteq \mathsf{D}$, consider the poset $(\mathsf{List}(A) \to \mathcal{P}(\mathsf{D}), \leq)$, where $Q \leq Q'$ if $Q(l) \subseteq Q'(l)$ for all l in $\mathsf{List}(A)$[3]. This poset forms a complete lattice, hence every monotone function on it has a least fixpoint. Given also a function P in $\mathsf{List}(A) \to \mathcal{P}(\mathsf{D})$, define $\mathsf{Bar}(A, P)$ as the least fixpoint of the following monotone endofunction $\Psi_{\mathsf{Bar}(A,P)}$ on $(\mathsf{List}(A) \to \mathcal{P}(\mathsf{D}), \leq)$:

$$\Psi_{\mathsf{Bar}(A,P)}(Q)(l) := \{\mathsf{base}\,X \mid X \in P(l)\} \cup \{\mathsf{step}\,Y \mid \forall a \in A.\, Y\,a \in Q\,(\mathsf{cons}\,a\,l)\}.$$

Finally, we introduce the DCPO $\mathcal{L}$ of pairs (S, F) with $S \subseteq \mathsf{D}$ and $F \in S \to \mathcal{P}(\mathsf{D})$ as described in Section 4.1. Here S is to be viewed as a set of types, and F as a function giving the set of elements $F(T) \subseteq \mathsf{D}$ for each $T \in S$. We are now ready to define which terms of Λ are types, and which elements each type has. We do so in two stages, first for the small types and then for all types, using monotone endofunctions Φ_0 and Φ_1 on $\mathcal{L}$, respectively.

An important observation is that only constructors play a role here, not operators, and that the only difference between Φ_0 and Φ_1 is that the latter includes the constructor U.

[3] Abusing notation, we denote by $A \to B$ the set of functions from the set A to the set B in the ambient naive set theory in which we develop the realizability model.

We define (T_0, El_0) in $\mathcal{L}$ to be the least fixpoint of the following monotone endofunction Φ_0 on $\mathcal{L}$:

$$\Phi_0(S, F) := (S_1 \cup S_2 \cup \cdots \cup S_9, F_1 \cup F_2 \cup \cdots \cup F_9)$$

where

$$\begin{array}{lcl}
S_1 & = & \{\mathsf{N}_0\} \\
F_1 & = & \{(\mathsf{N}_0, \emptyset)\} \\
S_2 & = & \{\mathsf{N}_1\} \\
F_2 & = & \{(\mathsf{N}_1, \{\mathsf{0}\})\} \\
S_3 & = & \{\mathsf{N}_2\} \\
F_3 & = & \{(\mathsf{N}_2, \{\mathsf{0}, \mathsf{1}\})\} \\
S_4 & = & \{\mathsf{N}\} \\
F_4 & = & \{(\mathsf{N}, \mathsf{Num})\} \\
S_5 & = & \{A + B \mid A, B \in S\} \\
F_5 & = & \{(A + B, \{\mathsf{inl}\, a \mid a \in F(A)\} \cup \{\mathsf{inr}\, b \mid b \in F(B)\}) \mid A, B \in S\} \\
S_6 & = & \{[A] \mid A \in S\} \\
F_6 & = & \{([A], \mathsf{List}(F(A))) \mid A \in S\} \\
S_7 & = & \{\Pi x{:}A.\, B \mid A \in S \wedge \forall a \in F(A).\, B(a) \in S\} \\
F_7 & = & \{(\Pi x{:}A.\, B, \{M \mid \forall a \in F(A).\, M\, a \in F(B(a))\}) \mid \Pi x{:}A.\, B \in S_7\} \\
S_8 & = & \{\Sigma x{:}A.\, B \mid A \in S \wedge \forall a \in F(A).\, B(a) \in S\} \\
F_8 & = & \{(\Sigma x{:}A.\, B, \{(a, M) \mid a \in F(A) \wedge M \in F(B(a))\}) \mid \Sigma x{:}A.\, B \in S_8\} \\
S_9 & = & \{\mathsf{bAr}\, P\, l \mid A \in S \wedge l \in \mathsf{List}(F(A)) \wedge \forall l' \in \mathsf{List}(F(A)).\, P\, l' \in S\} \\
F_9 & = & \{(\mathsf{bAr}\, P\, l, \mathsf{Bar}(F(A), \mathsf{List}(F(A)) \ni l' \mapsto F(P\, l'))(l)) \mid \mathsf{bAr}\, P\, l \in S_9\}
\end{array}$$

In the last line, the notation $\mathsf{List}(F(A)) \ni l' \mapsto F(P\, l')$ denotes the function which maps l' in $\mathsf{List}(F(A))$ to $F(P\, l')$.

The endofunction Φ_0 is monotone on the DCPO $\mathcal{L}$, hence the least fixpoint (T_0, El_0) exists by Theorem 9.

▶ **Definition 10.** (T_0, El_0) is the least fixpoint of Φ_0 above.

Given T_0, we define

$$\Phi_1(S, F) := (\{\mathsf{U}\} \cup S_1 \cup S_2 \cup \cdots \cup S_9, \{(\mathsf{U}, T_0)\} \cup F_1 \cup F_2 \cup \cdots \cup F_9)$$

with S_i, F_i for $i = 1, \ldots, 9$ are as defined earlier. The endofunction Φ_1 is monotone.

▶ **Definition 11.** (T_1, El_1) is the least fixpoint of Φ_1 above.

▶ Remark. An important observation about the definitions of Φ_0 and Φ_1 is that F occurs negatively in some clause, namely in S_7 for dependent products. This is the reason that we use DCPOs and not CPOs. Type-theoretically, the construction of the model goes by induction-recursion [5], as opposed to mutual induction. A different device, replacing negative occurrences by positive conditions on the complements, has been used in [1] and can be traced back to Scott and Feferman.

▶ Remark. The set T_1 is intended to contain all representatives of types in the type theory, both small and large, but it actually contains much more. Likewise, for $A \in T_1$, $El_1(A)$ intends to contain all interpretations of terms of type A in the type theory, but it actually contains much more. For instance, the set $El_1(\mathsf{N} \to \mathsf{N})$ contains all (total) recursive functions on Num. In particular, elements in T_1 or $El_1(A)$ (with $A \in T_1$) may not be normalizing. For instance, let

$$f := \lambda n.\, \mathsf{rec}\, 0\, (\lambda m.\, \lambda x.\, 0)\, n$$

and, deliberately using the term Ω which is not typable,

$$g := \lambda n.\, \mathsf{rec}\, 0\, \Omega\, (f\, n).$$

Then f always returns $\mathsf{0}$ on a numeral, i.e., $f\, n$ is $\beta\iota$-equivalent to $\mathsf{0}$ for any n in Num, so that also $gn =_{\beta\iota} \mathsf{0}$ for all numerals n. The term g is in $El_1(\mathsf{N} \to \mathsf{N})$, but is not even weakly normalizing, since g reduces to itself by a contraction of Ω. Likewise, for the term

$$h := \lambda n.\, (\mathsf{rec}\, \mathsf{N}\, \Omega\, (f\, n)$$

we have $hn =_{\beta\iota} \mathsf{N}$ for all numerals n. Hence $\mathsf{Pi}\, \mathsf{N}\, h$ is in T_0, but is not weakly normalizing. We have $f, g \in El_0(\mathsf{Pi}\, \mathsf{N}\, h) = El_0(\mathsf{N} \to \mathsf{N})$. While we have $\vdash f : \mathsf{N} \to \mathsf{N}$, it is neither possible to derive $\vdash g : \mathsf{N} \to \mathsf{N}$, nor $\vdash \mathsf{Pi}\, \mathsf{N}\, h$, let alone $\vdash f : \mathsf{Pi}\, \mathsf{N}\, h$ or $\vdash g : \mathsf{Pi}\, \mathsf{N}\, h$. The realizability model is not a term model. This is not a bug, but a feature that we will exploit: types that are inhabited in the model, can be consistently added as axioms to the type theory.

The following lemma states that El_0 and El_1 agree on T_0.

▶ **Lemma 12.** $(T_0, El_0) \leq (T_1, El_1)$.

Proof. The claim follows from Theorem 9, since (T_1, El_1) is a prefixpoint of Φ_0. ◀

From the fact that (T_1, El_1) (resp. (T_0, El_0)) is a fixpoint of Φ_1 (resp. Φ_0), we obtain the following lemma.

▶ **Lemma 13.** *For $i = 0, 1$, the following conditions hold:*

1) $\mathsf{N}_0 \in T_i$, *and* $El_i(\mathsf{N}_0) = \emptyset$*;*

2) $\mathsf{N}_1 \in T_i$, *and* $El_i(\mathsf{N}_1) = \{\mathsf{0}\}$*;*

3) $\mathsf{N}_2 \in T_i$, *and* $El_i(\mathsf{N}_2) = \{\mathsf{0}, \mathsf{1}\}$*;*

4) $\mathsf{N} \in T_i$, *and* $El_i(\mathsf{N}) = \mathsf{Num}$*;*

5) $A + B \in T_i$ *if* $A, B \in T_i$, *and then*

$$El_i(A + B) = \{\mathsf{inl}\, a \mid a \in El_i(A)\} \cup \{\mathsf{inr}\, b \mid b \in El_i(B)\}\,;$$

6) $[A] \in T_i$ *if* $A \in T_0$, *and then* $El_i([A]) = \mathsf{List}(El_0(A))$*;*

7) $\Pi x{:}A.\, B \in T_i$ *if* $A \in T_i$ *and* $B(a) \in T_i$ *for all* $a \in El_i(A)$, *and then*

$$El_i(\Pi x{:}A.\, B) = \{M \mid \forall a \in El_i(A), M\, a \in El_i(B(a))\}\,;$$

8) $\Sigma x{:}A.\, B \in T_i$ *if* $A \in T_i$ *and* $B(a) \in T_i$ *for all* $a \in El_i(A)$, *and then*

$$El_i(\Sigma x{:}A.\, B) = \{(W, P) \mid W \in El_i(A) \wedge P \in El_i(B(W))\}\,;$$

9) $\mathsf{bAr}\, P\, l \in T_i$ *if* $A \in T_0$, $l \in El_0([A])$ *and* $P\, l' \in T_0$ *for all* $l' \in El_0([A])$, *and then*

$$El_i(\mathsf{bAr}\, P\, l) = \mathsf{Bar}(El_0(A), El_0([A]) \ni l' \mapsto El_0(P\, l'))(l);$$

10) $\mathsf{U} \in T_1$, *and* $El_1(\mathsf{U}) = T_0$ *(but not* $\mathsf{U} \in T_0$*).*

4.3 Soundness

We now give the semantics of expressions and types a priori, that is, without any assumption of them being well-typed.

▶ **Definition 14.** An *environment* is a mapping from the set of variables to the domain D. We let $\rho, \rho', \ldots$ range over environments and let Env denote the set of all environments. By $\rho(x/a)$ we denote the environment ρ' with $\rho'(x) = a$ and $\rho'(y) = \rho(x)$ for variables $y \neq x$.

▶ **Definition 15.** Let M be a term, i.e., either an expression or a type, and ρ an environment. The *semantics* $[\![M]\!]_\rho \in \mathsf{D}$ of M in ρ denotes the ($\beta\iota$-equivalence class of the) result of the simultaneous substitution in M of all free occurrences of variables x by $\rho(x)$.

We write $[\![M]\!]$ to denote $[\![M]\!]_\rho$ with ρ being the empty environment, or when M is closed. We may also write M for $[\![M]\!]_\rho$ in that case.

As usual, we need a substitution lemma.

▶ **Lemma 16.** *For all M, N and ρ, we have $[\![(\lambda x.\, M)N]\!]_\rho = [\![M(N)]\!]_\rho = [\![M]\!]_{\rho(x/[\![N]\!]_\rho)}$.*

Proof. By a routine induction on the structure of M. ◀

We have to take into account certain sanity conditions on environments with respect to typing contexts.

▶ **Definition 17.** An environment ρ is Γ-*correct* if for all $(x : A) \in \Gamma$, $[\![A]\!]_\rho \in T_1$ and $\rho(x) \in El_1([\![A]\!]_\rho)$.

The following lemma states that the type theory is sound with respect to the semantics.

▶ **Lemma 18.** *For all Γ, M, A and for any Γ-correct ρ we have the following:*
1. *if $\Gamma \vdash A$, then $[\![A]\!]_\rho \in T_1$;*
2. *if $\Gamma \vdash M : A$, then $[\![A]\!]_\rho \in T_1$ and $[\![M]\!]_\rho \in El_1([\![A]\!]_\rho)$.*

Proof. Since the rules in Figure 1 mix (1) and (2), we prove the lemma by simultaneous induction on derivations. We start with the general typing rules in Figure 1.

Suppose $\Gamma \vdash \mathsf{U}$ by $\Gamma \vdash$, and ρ is Γ-correct. Then, the claim holds trivially since $[\![\mathsf{U}]\!]_\rho = \mathsf{U} \in T_1$ by Lemma 13.

Suppose $\Gamma \vdash A$ by $\Gamma \vdash A : \mathsf{U}$, and ρ is Γ-correct. We have $[\![A]\!]_\rho \in El_1(\mathsf{U})$ by induction hypothesis and $El_1(\mathsf{U}) = T_0 \subseteq T_1$ by Lemma 12, from which the claim follows.

Suppose $\Gamma \vdash \Pi x{:}A.\, B$ by $\Gamma \vdash A$ and $\Gamma, x : A \vdash B$, and ρ is Γ-correct. We have to prove that $[\![\Pi x{:}A.\, B]\!]_\rho \in T_1$. By induction hypothesis on $\Gamma \vdash A$, we have $[\![A]\!]_\rho \in T_1$. By Lemma 13 and an appeal to the Substitution Lemma, it suffices that $[\![B]\!]_{\rho(x/a)} \in T_1$ for all $a \in El_1([\![A]\!]_\rho)$. For this, it suffices by induction hypothesis on $\Gamma, x : A \vdash B$ that $\rho(x/a)$ is $(\Gamma, x : A)$-correct, which follows from $a \in El_1([\![A]\!]_\rho)$.

Suppose $\Gamma \vdash x : A$ by $(x : A) \in \Gamma$. Then the claim follows by that ρ is Γ-correct and $[\![x]\!]_\rho = \rho(x)$.

Suppose $\Gamma \vdash M : B$ by $A =_{\beta\iota} B$, $\Gamma \vdash M : A$ and $\Gamma \vdash B$. By induction hypothesis on $\Gamma \vdash M : A$, we have $[\![A]\!]_\rho \in T_1$ and $[\![M]\!]_\rho \in El_1([\![A]\!]_\rho)$. By $A =_{\beta\iota} B$, we have $[\![A]\!]_\rho = [\![B]\!]_\rho$, hence we get $[\![B]\!]_\rho \in T_1$ and $[\![M]\!]_\rho \in El_1([\![B]\!]_\rho)$[4]

[4] We do not use the induction hypothesis on $\Gamma \vdash B$. In fact, we do not use the hypothesis $\Gamma \vdash B$. This manifests that the typing rules are more restrictive than the semantics.

Suppose $\Gamma \vdash \lambda x.\, M : \Pi x{:}A.\, B$ by $\Gamma \vdash A$, and $\Gamma \vdash \Pi x{:}A.\, B$ and $\Gamma, x : A \vdash M : B$. By induction hypothesis, we have $[\![A]\!]_\rho \in T_1$ and $[\![\Pi x{:}A.\, B]\!]_\rho \in T_1$. We have to show $[\![\lambda x.\, M]\!]_\rho \in El_1([\![\Pi x{:}A.\, B]\!]_\rho)$. By the Substitution Lemma, it suffices that $[\![\lambda x.\, M]\!]_\rho\, a = [\![M]\!]_{\rho(x/a)} \in El_1([\![B]\!]_{\rho(x/a)})$ for all $a \in El_1([\![A]\!]_\rho)$. This follows from induction hypothesis on $\Gamma, x : A \vdash M : B$, noting that $\rho(x/a)$ is $(\Gamma, x : A)$-correct.

Suppose $\Gamma \vdash M\, N : B(N)$ by $\Gamma \vdash M : \Pi x{:}A.\, B$ and $\Gamma \vdash N : A$. By induction hypothesis, we have $[\![\Pi x{:}A.\, B]\!]_\rho \in T_1$, $[\![M]\!]_\rho \in El_1([\![\Pi x{:}A.\, B]\!]_\rho)$, and $[\![A]\!]_\rho \in T_1$, $[\![N]\!]_\rho \in El_1([\![A]\!]_\rho)$. By Lemma 13, it follows that $[\![M]\!]_\rho\, [\![N]\!]_\rho \in El_1([\![B]\!]_\rho([\![N]\!]_\rho))$. Using the Substitution Lemma, we conclude $[\![M\, N]\!]_\rho = [\![M]\!]_\rho\, [\![N]\!]_\rho \in El_1([\![B]\!]_\rho([\![N]\!]_\rho)) = El_1([\![B]\!]_{\rho(x/[\![N]\!]_\rho)}) = El_1([\![B(N)]\!]_\rho)$.

Suppose $\Gamma \vdash \Pi x{:}A.\, B : \mathsf{U}$ by $\Gamma \vdash A : \mathsf{U}$ and $\Gamma, x : A \vdash B : \mathsf{U}$. By induction hypothesis on $\Gamma \vdash A : \mathsf{U}$ and by Lemma 13, we know that $[\![\mathsf{U}]\!]_\rho = \mathsf{U} \in T_1$, and $[\![A]\!]_\rho \in El_1(\mathsf{U}) = T_0$. We have to show $[\![\Pi x{:}A.\, B]\!]_\rho \in T_0$. By Lemma 13 again and by the Substitution Lemma, it suffices that $[\![B]\!]_{\rho(x/a)} \in T_0$ for all $a \in El_0([\![A]\!]_\rho)$. Recalling that El_0 and El_1 agree on T_0 (Lemma 12), hence in particular on $[\![A]\!]_\rho$, this follows from induction hypothesis on $\Gamma, x : A \vdash B : \mathsf{U}$ since $\rho(x/a)$ is $(\Gamma, x : A)$-correct.

We are done with the general typing rules. We move on to the specific typing rules in Section 2.3. In the following, we will often tacitly use that $El_1(\mathsf{U}) = T_0 \subseteq T_1$ and that El_1 extends El_0 (Lemma 12).

Regarding the empty type, we have $\mathsf{U} \in T_1$ and $\mathsf{N}_0 \in El_1(\mathsf{U})$ from Lemma 13. If $\Gamma \vdash \mathsf{ExF} : \mathsf{N}_0 \to A$ by $\Gamma \vdash A : \mathsf{U}$, then by induction hypothesis, we have $[\![A]\!]_\rho \in T_0$. It follows that $\mathsf{N}_0 \to [\![A]\!]_\rho \in T_0$, and $\mathsf{ExF} \in El_0(\mathsf{N}_0 \to [\![A]\!]_\rho)$ since $El_0(\mathsf{N}_0)$ is empty.

Regarding the typing rules for the unit type, we get the claim from Lemma 13.

Regarding Booleans, we get the claim from Lemma 13 for the typing rules for N_2, $\mathsf{0}$ and $\mathsf{1}$. Suppose $\Gamma \vdash \mathsf{brec} : C\,\mathsf{0} \to C\,\mathsf{1} \to \Pi b{:}\mathsf{N}_2.\, C\, b$ by $\Gamma \vdash C : \mathsf{N}_2 \to \mathsf{U}$. By induction hypothesis, we have $\mathsf{N}_2 \to \mathsf{U} \in T_1$ and $[\![C]\!]_\rho \in El_1(\mathsf{N}_2 \to \mathsf{U})$, so $[\![C\,\mathsf{0}]\!]_\rho \in T_0$, $[\![C\,\mathsf{1}]\!]_\rho \in T_0$, and $[\![C\, b]\!]_\rho \in T_0$ for all $b \in El_1(\mathsf{N}_2)$. It follows that $[\![C\,\mathsf{0} \to C\,\mathsf{1} \to \Pi b{:}\mathsf{N}_2.\, C\, b]\!]_\rho \in T_1$. Let $M \in El_0([\![C\,\mathsf{0}]\!]_\rho)$ and $N \in El_0([\![C\,\mathsf{1}]\!]_\rho)$. For every $b \in El_1(\mathsf{N}_2)$, we have either $b = \mathsf{0}$ or $b = \mathsf{1}$. Hence we get $\mathsf{brec} \in El_1([\![C\,\mathsf{0} \to C\,\mathsf{1} \to \Pi b{:}\mathsf{N}_2.\, C\, b]\!]_\rho)$ from the ι-reduction for brec. We get the claim for the typing rule for beq from Lemma 13 and the ι-reduction for beq.

Regarding natural numbers, the only non-trivial rules are those for rec and eq. Suppose $\Gamma \vdash \mathsf{rec} : C\,\mathsf{0} \to (\Pi n{:}\mathsf{N}.\, C\, n \to C\,(S\, n)) \to \Pi n{:}\mathsf{N}.\, C\, n$ by $\Gamma \vdash C : \mathsf{N} \to \mathsf{U}$. By induction hypothesis, we have $\mathsf{N} \to \mathsf{U} \in T_1$ and $[\![C]\!]_\rho \in El_1(\mathsf{N} \to \mathsf{U})$. It follows that $[\![C\,\mathsf{0}]\!]_\rho \in T_0$, $[\![\Pi n{:}\mathsf{N}.\, C\, n \to C\,(S\, n)]\!]_\rho \in T_0$ and $[\![\Pi n{:}\mathsf{N}.\, C\, n]\!]_\rho \in T_0$, hence $[\![C\,\mathsf{0} \to (\Pi n{:}\mathsf{N}.\, C\, n \to C\,(S\, n)) \to \Pi n{:}\mathsf{N}.\, C\, n]\!]_\rho \in T_0$. We have to prove that $\mathsf{rec}\, M\, N\, n \in El_0([\![C\, n]\!]_\rho)$ for all $M \in El_0([\![C\,\mathsf{0}]\!]_\rho)$, $N \in El_0([\![\Pi n{:}\mathsf{N}.\, Cn \to C(Sn)]\!]_\rho)$ and $n \in El_0(\mathsf{N})$. This is proved by induction on $n \in El_0(\mathsf{N}) = \mathsf{Num}$, using the ι-rule for rec. It follows that $\mathsf{rec} \in El_0([\![C\,\mathsf{0} \to (\Pi n{:}\mathsf{N}.\, C\, n \to C\,(\mathsf{S}n)) \to \Pi n{:}\mathsf{N}.\, C\, n]\!]_\rho)$. Suppose $\Gamma \vdash \mathsf{eq} : \mathsf{N} \to \mathsf{N} \to \mathsf{U}$ by $\Gamma \vdash$. That $\mathsf{N} \to \mathsf{N} \to \mathsf{U} \in T_1$ follows from Lemma 13. In order to show $\mathsf{eq} \in El_1(\mathsf{N} \to \mathsf{N} \to \mathsf{U})$, we have to prove $\mathsf{eq}\, m\, n \in T_0$ for all m and n in Num. This is proved by nested induction on m and n.

Regarding lists, if $\Gamma \vdash [A] : \mathsf{U}$ by $\Gamma \vdash A : \mathsf{U}$, then by induction hypothesis we get $[\![A]\!]_\rho \in T_0$. It follows from Lemma 13 that $[\![[A]]\!]_\rho = [[\![A]\!]_\rho] \in T_0$. If $\Gamma \vdash \mathsf{nil} : [A]$ by $\Gamma \vdash A : \mathsf{U}$, the claim follows easily from the induction hypothesis and Lemma 13. The case for $\Gamma \vdash \mathsf{cons} : A \to [A] \to [A]$ by $\Gamma \vdash A : \mathsf{U}$ is similar. Suppose $\Gamma \vdash \mathsf{lrec} : C\,\mathsf{nil} \to (\Pi a{:}A.\, \Pi l{:}[A].\, C\, l \to C\,(a :: l)) \to \Pi l{:}[A].\, C\, l$ by $\Gamma \vdash A$ and $\Gamma \vdash C : [A] \to \mathsf{U}$. By induction hypothesis on $\Gamma \vdash C : [A] \to \mathsf{U}$, we have $[\![[A] \to \mathsf{U}]\!]_\rho \in T_1$ and $[\![C]\!]_\rho \in El_1([\![[A] \to \mathsf{U}]\!]_\rho)$. From Lemma 13, it follows that $[\![C\mathsf{nil}]\!]_\rho \in T_0$, $[\![\Pi a{:}A.\, \Pi l{:}[A].\, C\, l \to C\,(a :: l)]\!]_\rho \in T_1$ and $[\![\Pi l{:}[A].\, C\, l]\!]_\rho \in T_1$, hence $[\![C\,\mathsf{nil} \to (\Pi a{:}A.\, \Pi l{:}[A].\, C\, l \to C\,(a :: l)) \to \Pi l{:}[A].\, C\, l]\!]_\rho \in T_1$. In order to show that $\mathsf{lrec} \in El_1([\![C\,\mathsf{nil} \to (\Pi a{:}A.\, \Pi l{:}[A].\, C\, l \to C(a :: l)) \to \Pi l{:}[A].\, C\, l]\!]_\rho)$, we have to prove, for all $M \in El_1(\llbracket C\,\mathsf{nil}\rrbracket_\rho)$, $N \in El_1(\llbracket \Pi a{:}A.\,\Pi l{:}[A].\,C\,l \to C\,(a :: l)\rrbracket_\rho)$ and $l \in El_1([\llbracket A\rrbracket_\rho])$, that $\mathsf{lrec}\,M\,N\,l \in El_1(\llbracket C\,l\rrbracket_\rho)$. This is proved by induction on $l \in El_1([\llbracket A\rrbracket_\rho]) = \mathsf{List}(\llbracket A\rrbracket_\rho)$, using the ι-rewrite rules for lrec.

Sum types can be dealt with in a similar (but simpler) manner as list types.

Regarding dependent pairs, if $\Gamma \vdash \Sigma x{:}A.\,B : \mathsf{U}$ by $\Gamma \vdash A : \mathsf{U}$ and $\Gamma, x : A \vdash B : \mathsf{U}$, then we argue analogously to the case for $\Gamma \vdash \Pi x{:}A.\,B : \mathsf{U}$. Suppose $\Gamma \vdash (W, P) : \Sigma x{:}A.\,B$ by $\Gamma \vdash \Sigma x{:}A.\,B : \mathsf{U}$, $\Gamma \vdash W : A$ and $\Gamma \vdash P : B(W)$. We obtain $\Sigma x{:}A.\,B \in T_1$ by induction hypothesis. To show that $\llbracket (W,P)\rrbracket_\rho = (\llbracket W\rrbracket_\rho, \llbracket P\rrbracket_\rho) \in El_1(\llbracket \Sigma x{:}A.\,B\rrbracket_\rho)$, it suffices to prove $\llbracket W\rrbracket_\rho \in El_1(\llbracket A\rrbracket_\rho)$ and $\llbracket P\rrbracket_\rho \in El_1(\llbracket B\rrbracket_\rho(\llbracket W\rrbracket_\rho)) = El_1(\llbracket B(W)\rrbracket_\rho)$, both of which follow from induction hypothesis. Suppose $\Gamma \vdash \mathsf{srec} : (\Pi x{:}A.\,\Pi p{:}B.\,C\,(x,p)) \to \Pi y{:}(\Sigma x{:}A.\,B).\,C\,y$ by $\Gamma \vdash A : \mathsf{U}$, $\Gamma, x : A \vdash B : \mathsf{U}$, $\Gamma \vdash \Sigma x{:}A.\,B : \mathsf{U}$, and $\Gamma \vdash C : (\Sigma x{:}A.\,B) \to \mathsf{U}$. Induction hypotheses give us $\llbracket A\rrbracket_\rho \in T_0$, $\llbracket B\rrbracket_{\rho(x/a)} \in T_0$ for all $a \in El_0(\llbracket A\rrbracket_\rho)$, $\llbracket \Sigma x{:}A.\,B\rrbracket_\rho \in T_0$, and $\llbracket C\rrbracket_\rho \in El_1(\llbracket \Sigma x{:}A.\,B\rrbracket_\rho \to \mathsf{U})$. It follows that $\llbracket \Pi x{:}A.\,\Pi p{:}B.\,C\,(x,p)\rrbracket_\rho \in T_0$ and $\llbracket \Pi y{:}(\Sigma x{:}A.\,B).\,C\,y\rrbracket_\rho \in T_0$, and hence $\llbracket \Pi x{:}A.\,\Pi p{:}B.\,C\,(x,p) \to \Pi y{:}(\Sigma x{:}A.\,B).\,C\,y\rrbracket_\rho \in T_0$. By using the ι-rule for srec and by Lemma 13, we get $\mathsf{srec} \in El_0(\llbracket \Pi x{:}A.\,\Pi p{:}B.\,C\,(x,p) \to \Pi y{:}(\Sigma x{:}A.\,B).\,C\,y\rrbracket_\rho)$.

Regarding the constant bAr, if $\Gamma \vdash \mathsf{bAr} : ([A] \to \mathsf{U}) \to [A] \to \mathsf{U}$ by $\Gamma \vdash A : \mathsf{U}$, then we have $\llbracket ([A] \to \mathsf{U}) \to [A] \to \mathsf{U}\rrbracket_\rho \in T_1$ by induction hypothesis and Lemma 13. To show that $\mathsf{bAr} \in El_1(\llbracket ([A] \to \mathsf{U}) \to [A] \to \mathsf{U}\rrbracket_\rho) = El_1((\llbracket [A]\rrbracket_\rho \to \mathsf{U}) \to \llbracket [A]\rrbracket_\rho \to \mathsf{U})$, it suffices that $\mathsf{bAr}\,P\,l \in T_0$ for all $P \in El_1(\llbracket [A]\rrbracket_\rho \to \mathsf{U})$ and $l \in El_0(\llbracket [A]\rrbracket_\rho)$. This follows from Lemma 13, since $P\,l' \in T_0$ for all $l' \in El_0(\llbracket [A]\rrbracket_\rho)$. Suppose $\Gamma \vdash \mathsf{base}\,X : \mathsf{bAr}\,P\,l$ by $\Gamma \vdash A : \mathsf{U}$, $\Gamma \vdash l : [A]$, $\Gamma \vdash P : [A] \to \mathsf{U}$ and $\Gamma \vdash X : P\,l$. By induction hypothesis, we have $\llbracket A\rrbracket_\rho \in T_0$, $\llbracket l\rrbracket_\rho \in El_1(\llbracket [A]\rrbracket_\rho)$, and $\llbracket P\rrbracket_\rho \in El_1(\llbracket [A] \to \mathsf{U}\rrbracket_\rho) = El_1(\llbracket [A]\rrbracket_\rho \to \mathsf{U})$, hence $\llbracket P\rrbracket_\rho\,l' \in T_0$ for all $l' \in El_1(\llbracket [A]\rrbracket_\rho)$. This proves $\mathsf{bAr}\,P\,l \in T_0$. The induction hypothesis on $\Gamma \vdash X : P\,l$ gives us $\llbracket X\rrbracket_\rho \in El_1(\llbracket P\rrbracket_\rho\,\llbracket l\rrbracket_\rho)$, which proves $\llbracket \mathsf{base}\,X\rrbracket_\rho \in El_1(\llbracket \mathsf{bAr}\,P\,l\rrbracket_\rho)$ by Lemma 13. The case for $\Gamma \vdash \mathsf{step}\,Y : \mathsf{bAr}\,P\,l$ follows analogously. We will elaborate the last and most interesting case in some detail. Suppose

$$\begin{aligned}\Gamma \vdash \mathsf{barrec} : (\Pi l{:}[A].\,P\,l \to C\,l) \to \\ (\Pi l{:}[A].\,(\Pi a{:}A.\,C\,(a :: l)) \to C\,l) \to \Pi l{:}[A].\,\mathsf{bAr} P\,l \to C\,l\end{aligned}$$

by $\Gamma \vdash A : U$, $\Gamma \vdash P : [A] \to U$, and $\Gamma \vdash C : [A] \to U$. By induction hypothesis, using Lemma 13 many times, we get that the type of barrec is in T_1. In order to show that barrec is an element of the corresponding set

$$\begin{aligned}El_1((\llbracket \Pi l{:}[A].\,P\,l \to C\,l) \to \\ (\Pi l{:}[A].\,(\Pi a{:}A.\,C\,(a :: l)) \to C\,l) \to \Pi l{:}[A].\,\mathsf{bAr} P\,l \to C\,l\rrbracket_\rho),\end{aligned}$$

it suffices that

$$El_1(\mathsf{bAr}\,\llbracket P\rrbracket_\rho\,l) \subseteq \{M \in \mathsf{D} \mid \mathsf{barrec}\,B\,S\,l\,M \in El_1(\llbracket C\rrbracket_\rho\,l)\}$$

for all $B \in El_1(\llbracket \Pi l{:}[A].\,(P\,l \to C\,l)\rrbracket_\rho)$, $S \in El_1(\llbracket \Pi l{:}[A].\,((\Pi a{:}A.\,C\,(a :: l)) \to C\,l)\rrbracket_\rho)$, and $l \in El_1(\llbracket [A]\rrbracket_\rho)$. We prove this by fixpoint induction, recalling that, for fixed A, $El_1(\mathsf{bAr}\,\llbracket P\rrbracket_\rho\,l)$ is defined as the fixpoint of $\Psi_{\mathsf{Bar}(El_1(\llbracket A\rrbracket_\rho),\ El_1(\llbracket [A]\rrbracket_\rho) \ni l \mapsto El_1(\llbracket P\rrbracket_\rho l))}$. The latter operator will be abbreviated to Ψ, as A and P do not change in the proof.

We show that the function

$$\phi : El_1(\llbracket [A]\rrbracket_\rho) \ni l \mapsto \{M \in \mathsf{D} \mid \mathsf{barrec}\,S\,B\,l\,M \in El_1(\llbracket C\rrbracket_\rho\,l)\}$$

is a prefixpoint of Ψ, i.e., $\Psi(\phi)(l) \subseteq \phi(l)$ for all $l \in El_1(\llbracket [A]\rrbracket_\rho)$. By definition, there are two forms of elements in $\Psi(\phi)(l)$. The first is $\mathsf{base}\,X$ with $X \in El_1(\llbracket P\rrbracket_\rho\,l)$. Then we have

$\mathsf{barrec}\, B\, S\, l\, (\mathsf{base}\, X) = B\, l\, X \in El_1([\![C]\!]_\rho\, l$ by the assumptions on B and l. The second is $\mathsf{step}\, Y$ with, for all $a \in El_1([\![A]\!]_\rho)$, $Y a \in \phi\, (a :: l)$, that is, $\mathsf{barrec}\, B\, S\, (a :: l)\, (Y a) \in El_1([\![C]\!]_\rho\, (a :: l))$. Then we have $\mathsf{barrec}\, B\, S\, l\, (\mathsf{step}\, Y) = S\, l\, (\lambda a.\, \mathsf{barrec}\, B\, S\, (a :: l)\, (Y a) \in El_1([\![C]\!]_\rho\, l)$ by the assumptions on S and l.

It remains to prove that the auxiliary rules are sound. These rules define the type and computational behaviour of the constants $\mathsf{exists}, \mathsf{good}, \mathsf{length}, \mathsf{take}$.

Regarding the constant exists, if $\Gamma \vdash \mathsf{exists} : (A \to \mathsf{U}) \to [A] \to \mathsf{U}$ by $\Gamma \vdash A : \mathsf{U}$, then we have $[\![(A \to \mathsf{U}) \to [A] \to \mathsf{U}]\!]_\rho \in T_1$ by induction hypothesis and Lemma 13. To show that $\mathsf{exists} \in El_1([\![(A \to \mathsf{U}) \to [A] \to \mathsf{U}]\!]_\rho) = El_1(([\![A]\!]_\rho \to \mathsf{U}) \to [\![[A]]\!]_\rho \to \mathsf{U})$, it suffices that $\mathsf{exists}\, P\, l \in T_0$ for all $P \in El_1([\![A]\!]_\rho \to \mathsf{U})$ and $l \in \mathsf{List}(El_0([\![A]\!]_\rho))$. This follows by induction on l from the ι-reduction rules for exists. The argumentation for the other constants is very similar, and will hence be left to the reader. ◀

We obtain that if an expression M has a type A, then M realizes A.

▶ **Corollary 19.** *If $\vdash M : A$, then $A \in T_1$ and $M \in El_1(A)$.*

5 A Realizer for Markov's Principle

Markov's Principle is the following type:

$$\mathsf{MP} := \Pi f{:}\mathsf{N}{\to}\mathsf{N}_2.\, (\neg\neg\Sigma n{:}\mathsf{N}.\, \mathsf{beq}\, (f\, n)\, 1) \to \Sigma n{:}\mathsf{N}.\, \mathsf{beq}\, (f\, n)\, 1$$

Clearly $\vdash \mathsf{MP} : \mathsf{U}$, so $\mathsf{MP} \in T_1$ by Corollary 19. As a proposition, MP is classically true but unprovable in Heyting Arithmetic [14]; MP is not inhabited in our type theory. However, as we will show in this section, MP can be consistently added to the type theory: $El_1(\mathsf{MP})$ is non-empty. In other words, MP is true in the realizability model.

The realizer $R_{MP} \in El_1(\mathsf{MP})$ to be defined below essentially performs an unbounded search for an n such that $f\, n = 1$. This is possible since the computational system, based on untyped lambda calculus, is Turing complete. To prove that the search always finds such an n, we use Markov's Principle on the metalevel. This is possible since we are allowed to reason classically in the ambient naive set theory.

Recall that $\mathsf{beq} : \mathsf{N}_2 \to \mathsf{N}_2 \to \mathsf{U}$, $\mathsf{beq}\, 1\, 1 = \mathsf{N}_1$ and $0 : \mathsf{N}_1$. Let Y be any fixed point operator in the untyped lambda calculus, for example $Y := \lambda f.\, (\lambda x.\, f\, (x\, x))\, (\lambda x.\, f\, (x\, x))$. Then $Y\, F = F\, (Y\, F)$ for any F, in particular for

$$F := \lambda s\, f\, n.\, \mathsf{brec}\, (s\, f\, (\mathsf{S}\, n))\, (n, 0)\, (f\, n)$$

Then we have, for $\mathsf{search} := Y\, F$, that

$$\mathsf{search}\, f\, n = (F\, \mathsf{search})\, f\, n = \mathsf{brec}\, (\mathsf{search}\, f\, (\mathsf{S}\, n))\, (n, 0)\, (f\, n)$$

This means that $\mathsf{search}\, f\, 0$ performs the required search for the first n such that $f\, n = 1$. If n is found, $(n, 0)$ is returned, that is, the pair consisting of the numeral n and the proof term 0 of type $\mathsf{beq}\, (f\, n)\, 1$.

We define $R_{MP} = \lambda f\, p.\, \mathsf{search}\, f\, 0$ and it remains to prove $R_{MP} \in El_1(\mathsf{MP})$. Note that p does not occur in $\mathsf{search}\, f\, 0$. This is typical for realizability: realizers of negative statements carry no computational content, they only witness that the statement that is negated has no realizers. To show $R_{MP} \in El_1(\mathsf{MP})$, let $f \in El_1(\mathsf{N}{\to}\mathsf{N}_2)$ and $p \in El_1(\neg\neg\Sigma n{:}\mathsf{N}.\, \mathsf{beq}\, (f\, n)\, 1)$. We have to prove that $\mathsf{search}\, f\, 0 \in El_1(\Sigma n{:}\mathsf{N}.\, \mathsf{beq}\, (f\, n)\, 1)$. Towards a contradiction, assume the latter set is empty. Then, any term *foo* is in $El_1(\neg\Sigma n{:}\mathsf{N}.\, \mathsf{beq}\, (f\, n)\, 1)$. It follows that

$p\,foo \in El_1(\mathsf{N}_0)$. This is absurd, so $El_1(\Sigma n{:}\mathsf{N}.\,\mathsf{beq}\,(f\,n)\,1)$ is non-empty. Since it is decidable for any numeral n, whether or not $(n, 0) \in El_1(\Sigma n{:}\mathsf{N}.\,\mathsf{beq}\,(f\,n)\,1)$, it follows by Markov's Principle that there exists a pair $(n, 0) \in El_1(\Sigma n{:}\mathsf{N}.\,\mathsf{beq}\,(f\,n)\,1)$. Hence there is also such a pair with the smallest n, that is, $\mathsf{search}\,f\,0 \in El_1(\Sigma n{:}\mathsf{N}.\,\mathsf{beq}\,(f\,n)\,1)$. Note the role of p in the above argument: it serves to prove termination of the search but does not influence the actual outcome.

6 A Realizer for the Undecidability of the Halting Problem

The purpose of this section is to argue that, in addition to MP, we can consistently add the undecidablity of the halting problem to the type theory. Define

$$\mathsf{H_t} := \lambda n.\,\Sigma k{:}\mathsf{N}.\,\mathsf{beq}\,(t\,n\,n\,k)\,1$$

for $t : \mathsf{N} \to \mathsf{N} \to \mathsf{N} \to \mathsf{N}_2$ as described below. The intention is that $\mathsf{H_t}$ is a halting predicate, with t the characteristic function of Kleene's T-predicate [7, 2]. Using rec, we can define all primitive recursive functions, and actually many more. Since Kleene's T-predicate is primitive recursive, its characteristic function t is definable in the type theory. Kleene's T-predicate, $T\,e\,x\,w$, is based on a standard encoding of partial recursive functions as natural numbers. The first argument e of T is such a code of a partial recursive function, whereas the second argument x encodes an input to this function. The third argument w encodes a (terminating) computation sparked off by the function with code e on input with code x. Hence, $\mathsf{H_t}\,n$ holds if and only if the function encoded by n terminates on the input coded by n.

Let UH be the type:

$$\mathsf{UH} := \neg\Pi n{:}\mathsf{N}.\,(\mathsf{H_t}\,n + \neg\mathsf{H_t}\,n).$$

Clearly $\vdash \mathsf{UH} : \mathsf{U}$, so $\mathsf{UH} \in T_1$ by Corollary 19. We want to show that $El_1(\mathsf{UH})$ is non-empty. As UH is negative, it suffices to show that $\Pi n{:}\mathsf{N}.\,(\mathsf{H_t}\,n + \neg\mathsf{H_t}\,n)$ cannot be realized. Then any term realizes UH, so $El_1(\mathsf{UH})$ contains all terms of Λ (!). Towards a contradiction, assume $f \in El_1(\Pi n{:}\mathsf{N}.\,(H_t\,n + \neg H_t\,n))$. Diagonalizing over f we define:

$$d = \lambda n.\,\mathsf{case}\,\Omega\,1\,(f\,n)$$

such that in view or the definition of $\mathsf{H_t}$ we have for all $n : \mathsf{N}$:

$$d\,n = \Omega \iff f\,n = \mathsf{inl}(k, 0) \iff T(n, n, k),$$

where $(k, 0) \in El_1(\Sigma k{:}\mathsf{N}.\,\mathsf{beq}\,(t\,n\,n\,k)\,1)$.

As a lambda term, d represents a partial recursive function with code a numeral n_d[5]. Then we have $d\,n_d = \Omega \iff T(n_d, n_d, k)$ where $\mathsf{inl}(k, 0) = f\,n_d$, a plain contradiction with the choice of T and f. Therefore f as above cannot exist, and any term realizes UH. We conclude that UH can be consistently added to the type theory.

7 A set that is provably streamless but not provably Noetherian

In this section we shall prove that the converse of Theorem 5 is unprovable in type theory. The argument sketched in the introduction is that the converse is false when MP and UH

[5] The code n_d can in principle be constructed from d, but this is outside the scope of this paper.

are assumed. The unprovability in type theory then follows from the realizability model in which MP and UH are both valid. We start by some auxiliary definitions.

Given a predicate P on natural numbers, we define a binary relation $\overset{P}{=}$ to be the equality on the set of natural numbers n which satisfy P, irrelevant of the proof of $P\,n$. Formally, we define a typing rule for $\overset{P}{=}$ by

$$\frac{\Gamma \vdash P : \mathsf{N} \to \mathsf{U}}{\Gamma \vdash \overset{P}{=} : (\Sigma n{:}\mathsf{N}.\, P\,n) \to (\Sigma n{:}\mathsf{N}.\, P\,n) \to \mathsf{U}}$$

and ι-reduction, with $\overset{P}{=}$ written infix, given by

$$(n, h_n) \overset{P}{=} (m, h_m) = \mathsf{eq}\, n\, m.$$

Since eq is decidable, $\overset{P}{=}$ is decidable for any P.

Given a predicate P on natural numbers, we define a predicate $\overline{P}n$ to be true if Pk is decidable for all $k < n$. Formally, we define a typing rule for $\overline{P}$ by

$$\frac{\Gamma \vdash P : \mathsf{N} \to \mathsf{U}}{\Gamma \vdash \overline{P} : \mathsf{N} \to \mathsf{U}}$$

and ι-reductions give by

$$\begin{array}{lcl} \overline{P}0 & = & \mathsf{N}_1 \\ \overline{P}(\mathsf{S}n) & = & (P\,n + \neg P\,n) \times \overline{P}n \end{array}$$

The realizability model can easily be extended with sound interpretations of the above.

▶ **Lemma 20.** *There is a proof M such that*

$$P : \mathsf{N} \to \mathsf{U}, n : \mathsf{N} \vdash M : \overline{P}n \to \neg\neg\overline{P}(\mathsf{S}n).$$

Proof. We have to prove absurdity from $\overline{P}n$ and $\neg\overline{P}(\mathsf{S}n)$. Assume $Pn + \neg Pn$, then by $\overline{P}n$ we get $\overline{P}(\mathsf{S}n)$, which contradicts the assumption $\neg\overline{P}(\mathsf{S}n)$. Hence $\neg(Pn + \neg Pn)$, which is absurd, as $\neg\neg(A + \neg A)$ is a constructive tautology. ◀

▶ **Corollary 21.** *There is a proof M such that $P : \mathsf{N} \to \mathsf{U} \vdash M : \Pi n{:}\mathsf{N}.\, \neg\neg\overline{P}n$.*

Proof. By induction on n, using $\overline{P}0$ and basic facts about $\neg\neg$. ◀

Recall the terminology and notaion on decidability from Definition 1. We have the following easy lemmas about decidability.

▶ **Lemma 22.** *There exists proofs M_1, M_2, M_3, M_4 such that*

$$\begin{array}{r} A : \mathsf{U}, P : A \to \mathsf{U} \vdash M_1 : (\Pi x{:}A.\, \mathsf{dec}\,(P\,x)) \to \Pi l{:}[A].\, \mathsf{dec}\,(\mathsf{exists}\,P\,l) : \\ A : \mathsf{U}, R : A \to A \to \mathsf{U} \vdash M_2 : (\Pi x{:}A.\, \Pi y{:}A.\, \mathsf{dec}\,(R\,x\,y)) \to \Pi l{:}[A].\, \mathsf{dec}\,(\mathsf{good}\,R\,l); \\ P : \mathsf{N} \to \mathsf{U} \vdash M_3 : \Pi p_1{:}\Sigma n{:}\mathsf{N}.\, P\,n.\, \Pi p_2{:}\Sigma n{:}\mathsf{N}.\, P\,n.\, \mathsf{dec}\,(p_1 \overset{P}{=} p_2); \\ P : \mathsf{N} \to \mathsf{U} \vdash M_4 : \Pi l{:}[\Sigma n{:}\mathsf{N}.\, P\,n].\, \mathsf{dec}\,(\mathsf{good} \overset{P}{=} l). \end{array}$$

Proof. The first two are easily proved by induction on l, where the second uses the first. The third follows from the definition of $\overset{P}{=}$ and Lemma 2. The fourth follows from the second and the third. Note that the fourth typing states that it is decidable whether a list over a *subset* of natural numbers contains proof-irrelevant duplicates. ◀

In order to prove that the converse of Theorem 5 is not provable in the type theory, we construct a set which is provably not Noetherian, but can be proved streamless using MP and UH.

The following lemma is an abstract form of the argument in the introduction. In order to see this, recall that $(\mathsf{good}\overset{Q}{=})$ is a predicate expressing that a list contains a proof-irrelevant duplicate.

▶ **Lemma 23.** *Let A in bAr be the type $\Sigma n{:}\mathsf{N}.\,Q\,n$. There is a proof M such that*

$$Q : \mathsf{N} \to \mathsf{U} \vdash M : (\Pi n{:}\mathsf{N}.\,\neg\neg Q\,n) \to \Pi l{:}[\Sigma n{:}\mathsf{N}.\,Q\,n].\,\mathsf{bAr}\,(\mathsf{good}\overset{Q}{=})\,l \to \mathsf{good}\overset{Q}{=}l.$$

Proof. Let $Q : \mathsf{N} \to \mathsf{U}$ and $h_Q : \Pi n{:}\mathsf{N}.\,\neg\neg Q\,n$. We use induction on $\mathsf{bAr}\,(\mathsf{good}\overset{Q}{=})l$. If $\mathsf{bAr}\,(\mathsf{good}\overset{Q}{=})\,l$ by $\mathsf{good}\overset{Q}{=}l$, the claim holds immediately. Assume as induction hypothesis $\Pi x{:}(\Sigma n{:}\mathsf{N}.\,Q\,n).\,\mathsf{good}\overset{Q}{=}(x :: l)$. We have to prove $\mathsf{good}\overset{Q}{=}l$. By Lemma 22 we can reason by contradiction. Assume $\neg(\mathsf{good}\overset{Q}{=}l)$. We prove this is absurd and we are done. We perform case analysis on the shape of l. In case $l = \mathsf{nil}$, if $h_0 : Q\,0$, then $\mathsf{good}\overset{Q}{=}((0, h_0) :: \mathsf{nil})$ by the induction hypothesis. This is absurd, so $\neg Q\,0$, which is absurd by assumption h_Q. In case l is a non-empty list, let (n, h_n) be a maximum element in l, that is, for any (m, h_m) such that $\mathsf{exists}\,(\overset{Q}{=}(m, h_m))\,l$, we have that $n \geq m$. A maximum element exists since l is non-empty. It suffices to prove $\neg Q\,(\mathsf{S}\,n)$, which contradicts h_Q. Assume we have a proof $h_{\mathsf{S}\,n} : Q\,(\mathsf{S}\,n)$. By induction hypothesis, we have $\mathsf{good}\overset{Q}{=}((\mathsf{S}\,n, h_{\mathsf{S}\,n}) :: l)$. Since we assumed $\neg(\mathsf{good}\overset{Q}{=}l)$, it must be that $\mathsf{exists}\,(\overset{Q}{=}(\mathsf{S}\,n, h_{\mathsf{S}\,n}))\,l$, which contradicts with (n, h_n) being a maximum element in l. ◀

Noticing that it is absurd that the empty list is good, we deduce, from Lemma 23 and Corollary 21, that it is absurd that $\overset{H}{=}$ is Noetherian.

▶ **Lemma 24.** *There is a proof M such that $H : \mathsf{N} \to \mathsf{U} \vdash M : \neg(\mathsf{Noetherian}\overset{H}{=})$.*

On the other hand, in the presence of MP and UH, for $H_t : \mathsf{N} \to \mathsf{U}$ as defined in Section 6, we can prove that $\overset{H_t}{=}$ is streamless.

▶ **Lemma 25.** *There is a proof M such that*

$$\vdash M : \mathsf{MP} \to \mathsf{UH} \to \mathsf{streamless}\overset{\mathsf{H_t}}{=}.$$

Proof. Assume MP and UH. Given $f : \mathsf{N} \to \Sigma n{:}\mathsf{N}.\,\overline{\mathsf{H_t}}n$, we want to prove $\Sigma n{:}\mathsf{N}.\,\mathsf{good}\overset{\mathsf{H_t}}{=}(\overline{f}n)$. Noting that $(\mathsf{good}\overset{\mathsf{H_t}}{=})$ is decidable by Lemma 22, we can construct a function $e : \mathsf{N} \to \mathsf{N_2}$ such that $\mathsf{eq}\,(e\,n)\,1$ is true if and only if $\mathsf{good}\overset{\mathsf{H_t}}{=}(\overline{f}n)$ is true, for all $n : \mathsf{N}$. Thus, we may apply MP and it then suffices to prove $\neg\neg\Sigma n{:}\mathsf{N}.\,\mathsf{good}\overset{\mathsf{H_t}}{=}(\overline{f}n)$. Suppose $\neg\Sigma n{:}\mathsf{N}.\,\mathsf{good}\overset{\mathsf{H_t}}{=}(\overline{f}n)$, or equivalently, $\Pi n{:}\mathsf{N}.\,\neg\mathsf{good}\overset{\mathsf{H_t}}{=}(\overline{f}n)$. Then, for any given $n : \mathsf{N}$, the list $\overline{f}(\mathsf{S}(\mathsf{S}n))$ gives us a proof of $\mathsf{H_t}\,n + \neg\mathsf{H_t}\,n$[6]. This contradicts UH. ◀

[6] Informally, the list $\overline{f}(\mathsf{S}(\mathsf{S}n))$ contains a maximum element (m, h_m) such that $m \geq n$. The proof h_m of $\overline{\mathsf{H_t}}m$ gives us $H\,n + \neg H\,n$.

We have now shown that, in the presence of MP and UH, there is a relation that is streamless but cannot be Noetherian. Recalling that MP and UH are consistent with the type theory we work in, we conclude that it is unprovable in the type theory that every streamless relation is Noetherian.

▶ **Theorem 26.** *There is no proof M such that*

$$\vdash M : \mathsf{streamless} \overset{\overline{\mathsf{H_t}}}{=} \;\to\; \mathsf{Noetherian} \overset{\overline{\mathsf{H_t}}}{=}.$$

▶ **Corollary 27.** *There is no proof M such that*

$$\vdash M : \Pi A{:}\mathsf{U}.\,\Pi R{:}A \to A \to \mathsf{U}.\,\mathsf{streamless}\,R \;\to\; \mathsf{Noetherian}\,R.$$

8 Related work and concluding remarks

We have constructed a realizability model for Martin-Löf dependent type theory, viewed as a set of typing rules typing terms of an untyped lambda-calculus Λ. Similar realizability models have been given by Martin Löf [9] and Beeson [1]. We have paid extra attention to the details, in particular to those of the type of an inductive bar.

The purpose of the model is to demonstrate a particular unprovability result. It may be illuminating to discuss this result in connection with the well-known Kleene Tree [7, 2], a primitive recursive relation P on binary sequences which defines an infinite tree without an infinite recursive branch. The Kleene Tree is the prime example that Brouwer's Fan Theorem (an intuitionistic version of König's Lemma) fails in recursive analysis. In the context of this paper, Kleene's result yields that, with $A = \mathsf{N}_2$ and for a specific decidable $P : [\mathsf{N}_2] \to \mathsf{U}$, the following type is not inhabited:

$$(\Pi f{:}\mathsf{N} \to A.\,\Sigma n{:}\mathsf{N}.\,P\,(\overline{f}n)) \;\to\; \mathsf{bAr}\,P\,\mathsf{nil}.$$

For comparison, our result, with $A = \Sigma n{:}\mathsf{N}.\,\overline{\mathsf{H_t}}n$ and $Q := \mathsf{good} \overset{\overline{\mathsf{H_t}}}{=}$, states that the following type is not inhabited:

$$(\Pi f{:}\mathsf{N} \to A.\,\Sigma n{:}\mathsf{N}.\,Q\,(\overline{f}n)) \;\to\; \mathsf{bAr}\,Q\,\mathsf{nil}.$$

The important difference between the two results is the instantiation of the type A, that is, $A = \mathsf{N}_2$ for Kleene and $A = \Sigma n{:}\mathsf{N}.\,\overline{\mathsf{H_t}}n$ for us. Clearly, $A = \mathsf{N}_2$ is the simpler of the two. On the other hand, with $Ql := \mathsf{good} \overset{\overline{\mathsf{H_t}}}{=} l$ expressing that the list l contains a duplicate, we are allowed far less expressive power than Kleene's P. This explains to some extent why we use a more complicated base type $A = \Sigma n{:}\mathsf{N}.\,\overline{\mathsf{H_t}}n$, which is the simplest type we could find defining a subset of the natural numbers that is not finite in the sense of Noetherian, and at the same time finite in the sense of streamless in a consistent extension of the type theory. This confirms a conjecture formulated by Coquand and Spiwack in [4]. We conclude by formulating an open problem: does there exist a *decidable* P such that $\Sigma n{:}\mathsf{N}.\,Pn$ distinguishes between Noetherian and streamless?

References

1 M. Beeson. Recursive models for constructive set theories. *Annals of Mathematical Logic*, 23:127–178, 1982.

2 M. J. Beeson. *Foundations of Constructive Mathematics*, volume 6 of *Ergebnisse der Mathematik und ihrer Grenzgebiete*. Springer, 1985.

3 T. Coquand and A. Spiwack. A proof of strong normalisation using domain theory. *Logical Methods in Computer Science*, 3(4), 2007.

4 T. Coquand and A. Spiwack. Constructively finite? In L. Lambán, A. Romero, and J. Rubio, editors, *Contribuciones científicas en honor de Mirian Andrés Gómez*, pages 217–230. Publicaciones de la Universidad de La Rioja, 2010. ISBN 978-84-96487-50-5.

5 P. Dybjer. A general formulation of simultaneous inductive-recursive definitions in type theory. *Journal of Symbolic Logic*, 65(2):525–549, 2000.

6 M. Escardó. Joins in the frame of nuclei. *Applied Categorical Structures*, 11:117–124, 2003.

7 S.C. Kleene. Recursive functions and intuitionistic mathematics. In L.M. Graves, E. Hille, P.A. Smith, and O. Zariski, editors, *Proceedings of the International Congress of Mathematicians*, pages 679–685. AMS, 1952.

8 J. W. Klop, V. van Oostrom, and F. van Raamsdonk. Combinatory reduction systems: Introduction and survey. *Theoretical Computer Science*, 121(1&2):279–308, 1993.

9 P. Martin-Löf. An intuitionistic theory of types: predicative part. In H.E. Rose and J.C. Shepherdson, editors, *Logic Colloquium '73*, volume 80 of *Studies in Logic and the Foundations of Mathematics*, pages 73–118, Amsterdam, 1975. North-Holland.

10 E. Parmann. `https://github.com/epa095/noetherian-implies-streamless`.

11 E. Parmann. *Case Studies in Constructive Mathematics.* PhD thesis, University of Bergen, 2016.

12 D. Pataraia. A constructive proof of Tarski's fixed-point theorem for DCPOs. Presented at the 65th Peripatetic Seminar on Sheaves and Logic, November 1997.

13 C. Spector. Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles formulated in current intuitionistic mathematics. In J.C.E. Dekker, editor, *Recursive function theory, Proc. Symp. in pure mathematics V*, pages 1–27. AMS, 1962.

14 A. S. Troelstra, editor. *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*, volume 344 of *Lecture Notes in Mathematics*. Springer, 1973.

15 The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics.* The Univalent Foundations Program, Institute for Advanced Study, 2013. URL: `https://homotopytypetheory.org/book`.