Connecting Constructive Notions of Ordinals in Homotopy Type Theory

Nicolai Kraus
University of Nottingham, UK

Fredrik Nordvall Forsberg
University of Strathclyde, UK

Chuangjie Xu
fortiss GmbH, Germany

Abstract

In classical set theory, there are many equivalent ways to introduce ordinals. In a constructive setting, however, the different notions split apart, with different advantages and disadvantages for each. We consider three different notions of ordinals in homotopy type theory, and show how they relate to each other: A notation system based on Cantor normal forms, a refined notion of Brouwer trees (inductively generated by zero, successor and countable limits), and wellfounded extensional orders. For Cantor normal forms, most properties are decidable, whereas for wellfounded extensional transitive orders, most are undecidable. Formulations for Brouwer trees are usually partially decidable. We demonstrate that all three notions have properties expected of ordinals: their order relations, although defined differently in each case, are all extensional and wellfounded, and the usual arithmetic operations can be defined in each case. We connect these notions by constructing structure preserving embeddings of Cantor normal forms into Brouwer trees, and of these in turn into wellfounded extensional orders. We have formalised most of our results in cubical Agda.

2012 ACM Subject Classification Theory of computation → Type theory

Keywords and phrases Constructive ordinals, Cantor normal forms, Brouwer trees

Related Version This paper has appeared in the proceedings of the *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*. The MFCS version does not contain the appendix included here, but is otherwise essentially identical (including theorem numberings).

Supplementary Material A formalisation is available:
Agda source code: `https://bitbucket.org/nicolaikraus/constructive-ordinals-in-hott/src`
Agda as html: `https://cj-xu.github.io/agda/constructive-ordinals-in-hott/index.html`

Funding *Nicolai Kraus*: The Royal Society, grant reference URF\R1\191055.
Fredrik Nordvall Forsberg: UK National Physical Laboratory Measurement Fellowship project "Dependent types for trustworthy tools".
Chuangjie Xu: The Humboldt Foundation and the LMUexcellent program.

Acknowledgements We thank the participants of the conferences *Developments in Computer Science* and *TYPES*, as well as Helmut Schwichtenberg and Thorsten Altenkirch for fruitful discussions on this work. We are also grateful to the anonymous reviewers, whose remarks helped us improve the paper.

1 Introduction

The use of ordinals is a powerful tool when proving that processes terminate, when justifying induction and recursion [20, 24], or in (meta)mathematics generally. Unfortunately, the standard definition of ordinals is not very well-behaved constructively, meaning that additional work is required before this tool can be deployed in constructive mathematics or program verification tools based on constructive type theory such as Agda [33], Coq [15] or Lean [18].

Constructively, the classical notion of ordinal fragments into a number of inequivalent definitions, each with pros and cons. For example, “syntactic” ordinal notation systems [10, 36, 38] are popular with proof theorists, as their concrete character typically mean that equality and the order relation on ordinals are decidable. However, truly infinitary operations such as taking the limit of a countable sequence of ordinals are usually not constructible. We will consider a simple ordinal notation system based on Cantor normal forms [32], designed in such a way that there are no “junk” terms not denoting real ordinals.

Another alternative (based on notation systems by Church [14] and Kleene [28]), popular in the functional programming community, is to consider “Brouwer ordinal trees” $\mathcal{O}$ inductively generated by zero, successor and a “supremum” constructor

$$\mathsf{sup} : (\mathbb{N} \to \mathcal{O}) \to \mathcal{O}$$

which forms a new tree for every countable sequence of trees [8, 16, 26]. By the inductive nature of the definition, constructions on trees can be carried out by giving one case for zero, one for successors, and one for suprema, just as in the classical theorem of transfinite induction. However calling the constructor sup is wishful thinking; $\mathsf{sup}(s)$ does not faithfully represent the suprema of the sequence s, since we do not have that e.g. $\mathsf{sup}(s_0, s_1, s_2, \ldots) = \mathsf{sup}(s_1, s_0, s_2, \ldots)$ — each sequence gives rise to a new tree, rather than identifying trees representing the same suprema. We use the notion of higher inductive types [17, 30] from homotopy type theory [40] to remedy the situation and make a type of Brouwer trees which faithfully represents ordinals. Since our ordinals now can be infinitary, we lose decidability of equality and order relations, but we retain the possibility of classifying an ordinal as a zero, a successor or a limit.

One can also consider extensional wellfounded orders, a variation on the classical set-theoretical axioms more suitable for a constructive treatment [39], which was transferred to the setting of homotopy type theory in the HoTT book [40, Chapter 10], and significantly extended by Escardó [23]. One is then forced to give up most notions of decidability — it is not even possible to decide if a given ordinal is zero, a successor or a limit. However many operations can still be defined on such ordinals, and properties such as wellfoundedness can still be proven. This is also the notion of ordinal most closely related to the traditional notion, and thus the most obviously “correct” notion in a classical setting.

All in all, each of these approaches gives quite a different feel to the ordinals they represent: Cantor normal forms emphasise syntactic manipulations, Brouwer trees how every ordinal can be classified as a zero, successor or limit, and extensional wellfounded orders the set theoretic properties of ordinals. As a consequence, each notion of ordinals is typically used in isolation, with no interaction or opportunities to transfer constructions and ideas from one setting to another — e.g., do the arithmetic operations defined on Cantor normal forms obey the same rules as the arithmetic operations defined on Brouwer trees? The goal of this paper is to answer such questions by connecting together the different notions. We do this firstly by introducing an abstract axiomatic framework of what we expect of any notion of ordinal, and explore to what extent the notions above satisfy these axioms, and secondly by constructing faithful embeddings between the notions, which shows that they all represent a correct notion of ordinal from the point of view of classical set theory.

Contributions

- We identify an axiomatic framework for ordinals and ordinal arithmetic that we use to compare the situations above in the setting of homotopy type theory.
- We define arithmetic operations on Cantor normal forms [32] and prove them uniquely correct with respect to our abstract axiomatisation. This notion of correctness has not

been verified for Cantor normal forms previously, as far as we know.

- We construct a higher inductive-inductive type of Brouwer trees, and prove that their order is both wellfounded and extensional — properties which do not hold simultaneously for previous definitions of ordinals based on Brouwer trees. Further, we define arithmetic operations, and show that they are uniquely correct.
- We prove that the "set-theoretic" notion of ordinals [40, Section 10.3] satisfies our axiomatisation of addition and multiplication, and give constructive "taboos", showing that many operations on these ordinals are not possible constructively.
- We relate and connect these different notions of ordinals by constructing order preserving embeddings from more decidable notions into less decidable ones.

Formalisation and Full Proofs

We have formalised the material on Cantor normal forms and Brouwer trees in cubical Agda [43] at `https://cj-xu.github.io/agda/constructive-ordinals-in-hott/`; see also Escardó's formalisation [23] of many results on "set-theoretic" ordinals in HoTT. We have marked theorems with formalised and partly formalised proofs using the QED symbols ◀⚙⚙ and ◀⚙ respectively; they are also clickable links to the corresponding machine-checked statement. Moreover, pen-and-paper proofs for all our results can be found in the appendix.

Our formalisation uses the `{-# TERMINATING #-}` pragma to work around one known bug (issue #4725) and one limitation of the termination checker of Agda: recursive calls hidden under a propositional truncation are not seen to be structurally smaller. Such recursive calls when proving a proposition are justified by the eliminator presentation of [21] (although it would be non-trivial to reduce our mutual definitions to eliminators).

2 Underlying Theory and Notation

We work in and assume basic familarity with homotopy type theory (HoTT), i.e. Martin-Löf type theory extended with higher inductive types and the univalence axiom [40]. The central concept of HoTT is the Martin-Löf identity type, which we write as $a = b$ — we write $a \equiv b$ for definitional equality. We use Agda notation $(x : A) \to B(x)$ for the type of dependent functions, and write simply $A \to B$ if B does not depend on $x : A$. If the type in the domain can be inferred from context, we may simply write $\forall x.B(x)$ for $(x : A) \to B(x)$. Freely occurring variables are assumed to be $\forall$-quantified.

We denote the type of dependent pairs by $\Sigma(x : A).B(x)$, and its projections by fst and snd. We write $A \times B$ if B does not depend on $x : A$. We write $\mathcal{U}$ for a universe of types; we assume that we have a cumulative hierarchy $\mathcal{U}_i : \mathcal{U}_{i+1}$ of such universes closed under all type formers, but we will leave universe levels typically ambiguous.

We call a type A a *proposition* if all elements of A are equal, i.e. if $(x : A) \to (y : A) \to x = y$ is provable. We write $\mathsf{hProp} = \Sigma(A : \mathcal{U}).\mathsf{isProp}(A)$ for the type of propositions, and we implicitly insert a first projection if necessary, e.g. for $A : \mathsf{hProp}$, we may write $x : A$ rather than $x : \mathsf{fst}(A)$. A type A is a *set*, $A : \mathsf{hSet}$, if $(x = y) : \mathsf{hProp}$ for every $x, y : A$.

By $\exists(x : A).B(x)$, we mean the *propositional truncation* of $\Sigma(x : A).B(x)$, and if $(a, b) : \Sigma(x : A).B(x)$ then $|(a, b)| : \exists(x : A).B(x)$. The elimination rule of $\exists(x : A).B(x)$ only allows to define functions into propositions. By convention, we write $\exists k.P(k)$ for $\exists(k : \mathbb{N}).P(k)$. Finally, we write $A \uplus B$ for the sum type, $\mathbf{0}$ for the empty type, $\mathbf{1}$ for the type with exactly one element $*$, $\mathbf{2}$ for the type with two elements ff and tt, and $\neg A$ for $A \to \mathbf{0}$.

The *law of excluded middle* (LEM) says that, for every proposition P, we have $P \uplus \neg P$. Since we explicitly work with constructive notions of ordinals, we do not assume LEM, but rather use it as a *taboo*: a statement is not provable constructively if it implies LEM. Another, weaker, constructive taboo is the *weak limited principle of omniscience* WLPO: It says that any sequence $s : \mathbb{N} \to \mathbf{2}$ is either constantly ff, or it is not constantly ff.

3 Three Constructions of Types of Ordinals

We consider three concrete notions of ordinals in this paper, together with their order relations $<$ and $\leq$. The first notion is the one of *Cantor normal forms*, written Cnf, whose order is decidable. The second, written Brw, are *Brouwer Trees*, implemented as a higher inductive-inductive type. Finally, we consider the type Ord of ordinals that were studied in the HoTT book [40], whose order is undecidable, in general. In the current section, we briefly give the three definitions and leave the discussion of results for afterwards.

3.1 Cantor Normal Forms as a Subset of Binary Trees

In classical set theory, every ordinal α can be written uniquely in Cantor normal form

$$\alpha = \omega^{\beta_1} + \omega^{\beta_2} + \cdots + \omega^{\beta_n} \text{ with } \beta_1 \geq \beta_2 \geq \cdots \geq \beta_n \tag{1}$$

for some natural number n and ordinals β_i. If $\alpha < \varepsilon_0$, then $\beta_i < \alpha$, and we can represent α as a finite binary tree (with a condition) as follows [10, 12, 25, 32]. Let $\mathcal{T}$ be the type of unlabeled binary trees, i.e. the inductive type with suggestively named constructors $0 : \mathcal{T}$ and $\boldsymbol{\omega}^{-} \boldsymbol{+} - : \mathcal{T} \times \mathcal{T} \to \mathcal{T}$. Let the relation $<$ be the *lexicographical order*, i.e. generated by the following clauses:

$$0 < \boldsymbol{\omega}^{a} \boldsymbol{+} b$$
$$a < c \to \boldsymbol{\omega}^{a} \boldsymbol{+} b < \boldsymbol{\omega}^{c} \boldsymbol{+} d$$
$$b < d \to \boldsymbol{\omega}^{a} \boldsymbol{+} b < \boldsymbol{\omega}^{a} \boldsymbol{+} d.$$

We have the map $\mathsf{left} : \mathcal{T} \to \mathcal{T}$ defined by $\mathsf{left}(0) :\equiv 0$ and $\mathsf{left}(\boldsymbol{\omega}^{a} \boldsymbol{+} b) :\equiv a$ which gives us the left subtree (if it exists) of a tree. A tree is a *Cantor normal form* (CNF) if, for every $\boldsymbol{\omega}^{s} \boldsymbol{+} t$ that the tree contains, we have $\mathsf{left}(t) \leq s$, where $s \leq t :\equiv (s < t) \uplus (s = t)$; this enforces the condition in (1). For instance, both trees $1 :\equiv \boldsymbol{\omega}^{0} \boldsymbol{+} 0$ and $\omega :\equiv \boldsymbol{\omega}^{1} \boldsymbol{+} 0$ are CNFs. Formally, the predicate isCNF is defined inductively by the two clauses

$$\mathsf{isCNF}(0)$$
$$\mathsf{isCNF}(s) \to \mathsf{isCNF}(t) \to \mathsf{left}(t) \leq s \to \mathsf{isCNF}(\boldsymbol{\omega}^{s} \boldsymbol{+} t).$$

We write $\mathsf{Cnf} :\equiv \Sigma(t : \mathcal{T}).\mathsf{isCNF}(t)$ for the type of Cantor normal forms. We often omit the proof of $\mathsf{isCNF}(t)$ and call the tree t a CNF if no confusion is caused.

3.2 Brouwer Trees as a Quotient Inductive-Inductive Type

As discussed in the introduction, *Brouwer ordinal trees* (or simply *Brouwer trees*) are in functional programming often inductively generated by the usual constructors of natural numbers (*zero* and *successor*) and a constructor that gives a Brouwer tree for every sequence of Brouwer trees. To state a refined (*correct* in a sense that we will make precise and prove) version, we need the following notions:

Let A be a type and $\prec : A \to A \to \mathsf{hProp}$ be a binary relation. If f and g are two sequences $\mathbb{N} \to A$, we say that f is *simulated by* g, written $f \precsim g$, if $f \precsim g :\equiv \forall k.\exists n.f(k) \prec g(n)$. We say that f and g are *bisimilar* with respect to $\prec$, written $f \approx^{\prec} g$, if we have both $f \precsim g$ and $g \precsim f$. A sequence $f : \mathbb{N} \to A$ is *increasing* with respect to $\prec$ if we have $\forall k. f(k) \prec f(k+1)$. We write $\mathbb{N} \xrightarrow{\prec} A$ for the type of $\prec$-increasing sequences. Thus an increasing sequence f is a pair $f \equiv (\overline{f}, p)$ with p witnessing that $\overline{f}$ is increasing, but we keep the first projection implicit and write $f(k)$ instead of $\overline{f}(k)$.

Our type of Brouwer trees is a *quotient inductive-inductive type* [2], where we simultaneously construct the type $\mathsf{Brw} : \mathsf{hSet}$ together with a relation $\leq : \mathsf{Brw} \to \mathsf{Brw} \to \mathsf{hProp}$. The constructors for Brw are

$$\begin{aligned}
&\mathsf{zero}\ : \mathsf{Brw} \\
&\mathsf{succ}\ : \mathsf{Brw} \to \mathsf{Brw} \\
&\mathsf{limit} : (\mathbb{N} \xrightarrow{\leq} \mathsf{Brw}) \to \mathsf{Brw} \\
&\mathsf{bisim}\colon (f\, g : \mathbb{N} \xrightarrow{\leq} \mathsf{Brw}) \to f \approx^{\leq} g \to \mathsf{limit}\, f = \mathsf{limit}\, g,
\end{aligned}$$

where we write $x < y$ for $\mathsf{succ}\, x \leq y$ in the type of limit. Simulations thus use $\leq$ and the *increasing* predicate uses $<$, as one would expect. The truncation constructor, ensuring that Brw is a set, is kept implicit in the paper (but is explicit in the Agda formalisation).

The constructors for $\leq$ are the following, where each constructor is implicitly quantified over the variables $x, y, z : \mathsf{Brw}$ and $f : \mathbb{N} \xrightarrow{\leq} \mathsf{Brw}$ that it contains:

$$\begin{aligned}
&\leq\text{-}\mathsf{zero} &&: \mathsf{zero} \leq x \\
&\leq\text{-}\mathsf{trans} &&: x \leq y \to y \leq z \to x \leq z \\
&\leq\text{-}\mathsf{succ\text{-}mono} &&: x \leq y \to \mathsf{succ}\, x \leq \mathsf{succ}\, y \\
&\leq\text{-}\mathsf{cocone} &&: (k : \mathbb{N}) \to x \leq f(k) \to x \leq \mathsf{limit}\, f \\
&\leq\text{-}\mathsf{limiting} &&: (\forall k. f(k) \leq x) \to \mathsf{limit}\, f \leq x
\end{aligned}$$

The truncation constructor, which ensures that $x \leq y$ is a proposition, is again kept implicit.

We hope that the constructors of Brw and $\leq$ are self-explanatory. $\leq$-cocone ensures that $\mathsf{limit}\, f$ is indeed an upper bound of f, and $\leq$-$\mathsf{limiting}$ witnesses that it is the *least* upper bound or, from a categorical point of view, the (co)limit of f.

By restricting to limits of increasing sequences, we can avoid multiple representations of the same ordinal (as otherwise e.g. $a = \mathsf{limit}\,(\lambda\_.a)$). It is possible to drop this restriction, if one also strengthens the bisim constructor to witness antisymmetry — however we found this version of Brw significantly harder to work with.

3.3 Extensional Wellfounded Orders

The third notion of ordinals that we consider is the one studied in the HoTT book [40]. This is the notion which is closest to the classical definition of an ordinal as a set with a trichotomous, wellfounded, and transitive order, without a concrete representation. Requiring trichotomy leads to a notion that makes many constructions impossible in a setting where the law of excluded middle is not assumed. Therefore, when working constructively, it is better to replace the axiom of trichotomy by *extensionality*.

Concretely, an ordinal in the sense of [40, Def 10.3.17] is a type[1] X together with a relation $\prec : X \to X \to \mathsf{hProp}$ which is *transitive*, *extensional* (any two elements of X with the

[1] Note that [40, Def 10.3.17] asks for X to be a set, but this follows from the rest of the definition and we therefore drop this requirement.

same predecessors are equal), and *wellfounded* (every element is accessible, where accessibility is the least relation such that x is accessible if every predecessor of x is accessible.) — we will recall the precise definitions in Section 4. We write Ord for the type of ordinals in this sense. Note the shift of universes that happens here: the type Ord_i of ordinals with $X : \mathcal{U}_i$ is itself in $\mathcal{U}_{i+1}$. We are mostly interested in Ord_0, but note that Ord_0 lives in $\mathcal{U}_1$, while Cnf and Brw both live in $\mathcal{U}_0$.

We also have a relation on Ord itself. Following [40, Def 10.3.11 and Cor 10.3.13], a *simulation* between ordinals $(X, \prec_X)$ and $(Y, \prec_Y)$ is a function $f : X \to Y$ such that:
(a) f is monotone: $(x_1 \prec_X x_2) \to (f\, x_1 \prec_Y f\, x_2)$; and
(b) for all $x : X$ and $y : Y$, if $y \prec_Y f\, x$, then we have an $x_0 \prec_X x$ such that $f\, x_0 = y$.
We write $X \leq Y$ for the type of simulations between $(X, \prec_X)$ and $(Y, \prec_Y)$. Given an ordinal $(X, \prec)$ and $x : X$, the *initial segment* of elements below x is given as $X_{/x} :\equiv \Sigma(y : X).y \prec x$. Following [40, Def 10.3.19], a simulation $f : X \leq Y$ is *bounded* if we have $y : Y$ such that f induces an equivalence $X \simeq Y_{/y}$. We write $X < Y$ for the type of bounded simulations. This completes the definition of Ord together with type families $\leq$ and $<$.

4 An Abstract Axiomatic Framework for Ordinals

Which properties do we expect a type of ordinals to have? In this section, we go up one level of abstraction. We consider a type A with type families $<$ and $\leq : A \to A \to \mathcal{U}$, and discuss the properties that A with $<$ and $\leq$ can have. In Section 3, we introduced each of the types Cnf, Brw, and Ord together with its relations $<$ and $\leq$. Note that $\leq$ is the reflexive closure of $<$ in the case of Cnf, but for Brw and Ord, this is not constructively provable. In this section, we consider which properties they satisfy.

4.1 General Notions

A is a *set* if it satisfies the principle of unique identity proofs, i.e. if every identity type $a = b$ with $a, b : A$ is a proposition. Similarly, $<$ and $\leq$ are *valued in propositions* if every $a < b$ and $a \leq b$ is a proposition. A relation $<$ is *reflexive* if we have $a < a$, *irreflexive* if it is pointwise not reflexive $\neg(a < a)$, *transitive* if $a < b \to b < c \to a < c$, and *antisymmetric* if $a < b \to b < a \to a = b$. Further, the relation $<$ is *connex* if $(a < b) \uplus (b < a)$ and *trichotomous* if $(a < b) \uplus (a = b) \uplus (b < a)$.

▶ **Theorem 1.** *Each of* Cnf*,* Brw*, and* Ord *is a set, and their relations* $<$ *and* $\leq$ *are all valued in propositions. In each case, both* $<$ *and* $\leq$ *are transitive,* $<$ *is irreflexive, and* $\leq$ *is reflexive and antisymmetric. For* Cnf*, the relation* $<$ *is trichotomous and* $\leq$ *connex; for* Ord*, these statements are equivalent to the law of excluded middle.* ◀⚙

Proving that $\leq$ for Brw is antisymmetric is challenging because of the path constructors in the inductive-inductive definition of Brouwer trees. Antisymmetry and other technical properties discussed below require us to characterise the relation $\leq$ more explicitly, using an encode-decode argument [29]. By induction on x and y, we define the family Code such that $(\mathsf{Code}\, x\, y) \leftrightarrow (x \leq y)$. The cases for point constructors are unsurprising; for example, we define

$$\mathsf{Code}\,(\mathsf{limit}\, f)\,(\mathsf{succ}\, y) :\equiv \forall k.\mathsf{Code}\,(f\, k)\,(\mathsf{succ}\, y)$$
$$\mathsf{Code}\,(\mathsf{limit}\, f)\,(\mathsf{limit}\, g) :\equiv \forall k.\exists n.\mathsf{Code}\,(f\, k)\,(g\, n) \ .$$

The difficult part is defining Code for the path constructor bisim. If for example we have $g \approx h$, we need to show that $\mathsf{Code}\ (\mathsf{limit}\, f)\,(\mathsf{limit}\, g) \ = \ \mathsf{Code}\ (\mathsf{limit}\, f)\,(\mathsf{limit}\, h)$. The core argument is easy; using the bisimulation $g \approx h$, one can translate between indices for g and h with the appropriate properties. However, this example already shows why this becomes tricky: The bisimulation gives us inequalities ($\leq$), but the translation requires instances of Code, which means that toCode : $(x \leq y) \to (\mathsf{Code}\, x\, y)$ has to be defined *mutually* with Code. This is still not sufficient: In total, the mutual higher inductive-inductive construction needs to simultaneously prove and construct Code, toCode, versions of transitivity and reflexivity of Code as well several auxiliary lemmas. The complete definition is presented in the Agda formalisation (file `BrouwerTree.Code`). Once the definition of Code is shown correct, many technical properties are simple consequences.

From now on, we will assume that A is a set and that $<$ and $\leq$ are valued in propositions.

4.2 Extensionality and Wellfoundedness

Following [40, Def 10.3.9], we call a relation $<$ *extensional* if, for all $a, b : A$, we have $(\forall c. c < a \leftrightarrow c < b) \to b = a$, where $\leftrightarrow$ denotes "if and only if" (functions in both directions). Extensionality of $<$ for Brw is true, but non-trivial – note that it fails for the "naive" version of Brw, where the path constructor bisim is missing.

▶ **Theorem 2.** *For each of* Cnf*,* Brw*,* Ord*, both $<$ and $\leq$ are extensional.* ◀

We use the inductive definition of accessibility and wellfoundedness (with respect to $<$) by Aczel [1]. Concretely, the type family $\mathsf{acc} : A \to \mathcal{U}$ is inductively defined by the constructor

$$access : (a : A) \to ((b : A) \to b < a \to \mathsf{acc}(b)) \to \mathsf{acc}(a).$$

An element $a : A$ is called *accessible* if $\mathsf{acc}(a)$, and $<$ is *wellfounded* if all elements of A are accessible. It is well known that the following induction principle can be derived from the inductive presentation [40]:

▶ **Lemma 3** (Transfinite Induction). *Let $<$ be wellfounded and $P : A \to \mathcal{U}$ be a type family such that $\forall a.(\forall b < a.P(b)) \to P(a)$. Then, it follows that $\forall a.P(a)$.* ◀

In turn, transfinite induction can be used to prove that there is no infinite decreasing sequence if $<$ is wellfounded: $\neg\,(\Sigma(f : \mathbb{N} \to A).(i : \mathbb{N}) \to f(i+1) < f(i))$. A direct corollary is that if $<$ is wellfounded and valued in propositions, then its reflexive closure $(x < y) \uplus (x = y)$ is also valued in propositions, as $b < a$ and $b = a$ are mutually exclusive propositions.

▶ **Theorem 4.** *For each of* Cnf*,* Brw*,* Ord*, the relation $<$ is wellfounded.* ◀

The proof for Brw again makes crucial use of our encode-decode characterisation of $\leq$. Whenever $x < \mathsf{limit}\, f$, we can use the characterisation to find an $n : \mathbb{N}$ such that $x < f(n)$, which allows us to proceed with an inductive proof of wellfoundedness. Note that the results stated so far in particular mean that Cnf and Brw can be seen as elements of Ord themselves.

4.3 Classification as Zero, a Successor, or a Limit

All standard formulations of ordinals allow us to determine a minimal ordinal *zero* and (constructively) calculate the *successor* of an ordinal, but only some allows us to also calculate the *supremum* or *limit* of a collection of ordinals.

4.3.1 Assumptions

We have so far not required a relationship between $<$ and $\leq$, but we now need to do so in order for the concepts we define to be meaningful. We assume:

(A1) $<$ is transitive and irreflexive;
(A2) $\leq$ is reflexive, transitive, and antisymmetric;
(A3) we have $(<) \subseteq (\leq)$ and $(< \circ \leq) \subseteq (<)$.

The third condition 3 means that $(b < a) \to (b \leq a)$ and $(c < b) \to (b \leq a) \to (c < a)$. The "symmetric" variation

$$(\leq \circ <) \subseteq (<)$$

is true for Cnf and Brw, but for Ord, it is equivalent to the law of excluded middle — hence, we do not assume it. This constructive failure is known, and can be seen as motivation for *plump* ordinals [39, 37]. Of course, the above assumptions are satisfied if $\leq$ is the reflexive closure of $<$, but we again emphasise that this is not necessarily the case.

▶ **Theorem 5.** *For each of* Cnf*,* Brw*,* Ord*, assumptions 1 to 3 are satisfied.* ◀

For the remaining concepts, we assume that $<$ and $\leq$ satisfy the discussed assumptions.

4.3.2 Zero and (Strong) Successors

Let a be an element of A. It is *zero*, or *bottom*, if it is at least as small as any other element

$$\textsf{is-zero}(a) :\equiv \forall b. a \leq b, \tag{2}$$

and we say that the triple $(A, <, \leq)$ *has a zero* if we have an inhabitant of the type $\Sigma(z : A).\textsf{is-zero}(z)$. Both the types "being a zero" and "having a zero" are propositions.

▶ **Theorem 6.** Cnf*,* Brw*,* Ord *each have a zero.* ◀

We say that a is a *successor* of b if it is the least element strictly greater[2] than b:

$$(a \textsf{ is-suc-of } b) :\equiv (b < a) \times \forall x > b. x \geq a.$$

We say that $(A, <, \leq)$ *has successors* if there is a function $s : A \to A$ which calculates successors, i.e. such that $\forall b. s(b) \textsf{ is-suc-of } b$. "Calculating successors" and "having successors" are propositional properties, i.e. if a function that calculates successors exists, then it is unique. The following statement is simple but useful. Its proof uses assumption 3.

▶ **Lemma 7.** *Let $s : A \to A$ be given. The function s calculates successors if and only if $\forall bx. (b < x) \leftrightarrow (s\, b \leq x)$.* ◀

Dual to "a is the least element strictly greater than b" is the statement that "b is the greatest element strictly below a", in which case it is natural to call b the *predecessor* of a. If a is the successor of b and b the predecessor of a, then we call a the *strong successor* of b:

$$a \textsf{ is-str-suc-of } b :\equiv a \textsf{ is-suc-of } b \times \forall x < a. x \leq b.$$

We say that A *has strong successors* if there is $s : A \to A$ which calculates strong successors, i.e. such that $\forall b. s(b) \textsf{ is-str-suc-of } b$. The additional information contained in a strong successor play an important role in our technical development. A function $f : A \to A$ is $<$-*monotone* or $\leq$-*monotone* if it preserves the respective relation.

[2] Note that $>$ and $\geq$ are the obvious symmetric notations for $<$, $\leq$; they are *not* newly assumed relations.

▶ **Theorem 8.** *Each of the three types* Cnf*,* Brw*,* Ord *has strong successors. The successor functions of* Cnf *and* Brw *are both $<$- and $\leq$-monotone. For the successor function of* Ord*, either monotonicity property is equivalent to the law of excluded middle.* ◀⚙

For Cnf, the successor function is given by adding a leaf, for Brw by the constructor with the same name, and for Ord, one forms the coproduct with the unit type.

4.3.3 Suprema and Limits

Finally, we consider *suprema/least upper bounds* of $\mathbb{N}$-indexed sequences. We say that a is a *supremum* or the *least upper bound* of $f : \mathbb{N} \to A$, if a is at least as large as every f_i, and if any other x with this property is at least as large as a:

$$(a \;\textsf{is-sup-of}\; f) :\equiv (\forall i. f_i \leq a) \times (\forall x. (\forall i. f_i \leq x) \to a \leq x).$$

We say that $(A, <, \leq)$ *has suprema* if there is a function $\sqcup : (\mathbb{N} \to A) \to A$ which calculates suprema, i.e. such that $(f : \mathbb{N} \to A) \to (\sqcup f) \;\textsf{is-sup-of}\; f$. The supremum of a sequence is unique if it exists, i.e. the type of suprema is propositional for a given sequence f. Both the properties "calculating suprema" and "having suprema" are propositions.

Every $a : A$ is trivially the supremum of the sequence constantly a, and therefore, "being a supremum" does not describe the usual notion of *limit ordinals.* One might consider a a *proper* supremum of f if a is pointwise strictly above f, i.e. $\forall i. f_i < a$. This is automatically guaranteed if f is increasing with respect to $<$, and in this case, we call a the *limit* of f:

$$\begin{aligned} &\_ \;\textsf{is-lim-of}\; \_ : A \to (\mathbb{N} \xrightarrow{<} A) \to \mathcal{U} \\ &a \;\textsf{is-lim-of}\; (f, q) :\equiv a \;\textsf{is-sup-of}\; f. \end{aligned}$$

We say that A *has limits* if there is a function $\textsf{limit} : (\mathbb{N} \xrightarrow{<} A) \to A$ that calculates limits.

Note that Cnf cannot have limits since one can construct a sequence (see Theorem 22) which comes arbitrarily close to ε_0. This motivates the restriction to *bounded* sequences, i.e. a sequence f with a b : Cnf such that $f_i < b$ for all i.

▶ **Theorem 9.** Cnf *does not have suprema or limits.* Brw *has limits of increasing sequences by construction.* Ord *also has limits of increasing sequences, and moreover limits of* weakly *increasing sequences (i.e. sequences increasing with respect to $\leq$).*

Assuming the law of exclude middle, Cnf *has suprema (and thus limits) of arbitrary* bounded *sequences. If* Cnf *has limits of bounded increasing sequences, then the weak limited principle of omniscience (WLPO) is derivable.* ◀⚙

We expect that it is not constructively possible to calculate suprema (or even binary joins) in Brw, as it seems this would make it possible to decide if a limit reaches past $\omega + 1$ or not, which is a constructive taboo.

4.3.4 Classifiability

For classical set-theoretic ordinals, every ordinal is either zero, a successor, or a limit. We say that a notion of ordinals which allows this is has classification. This is very useful, as many theorems that start with "for every ordinal" have proofs that consider the three cases separately. In the same way as not all definitions of ordinals make it possible to calculate limits, only some formulations make it possible to constructively classify any given ordinal.

We already defined what it means to be a zero in (2). We now also define what it means for $a : A$ to be a strong successor or a limit:

$$\mathsf{is\text{-}str\text{-}suc}(a) :\equiv \Sigma(b : A).(a \mathrel{\mathsf{is\text{-}str\text{-}suc\text{-}of}} b) \qquad \mathsf{is\text{-}lim}(a) :\equiv \exists f : \mathbb{N} \to A.a \mathrel{\mathsf{is\text{-}lim\text{-}of}} f.$$

All of $\mathsf{is\text{-}zero}(a)$, $\mathsf{is\text{-}str\text{-}suc}(a)$ and $\mathsf{is\text{-}lim}(a)$ are propositions; note that this is true even though $\mathsf{is\text{-}str\text{-}suc}(a)$ is defined without a propositional truncation.

▶ **Lemma 10.** *Any $a : A$ can be at most one out of {zero, strong successor, limit}, and in a unique way. In other words, the type $\mathsf{is\text{-}zero}(a) \uplus \mathsf{is\text{-}str\text{-}suc}(a) \uplus \mathsf{is\text{-}limit}(a)$ is a proposition.* ◀

We say that an element of A is *classifiable* if it is zero or a strong successor or a limit. We say $(A, <, \leq)$ *has classification* if every element of A is classifiable. By Lemma 10, $(A, <, \leq)$ has classification exactly if the type $\mathsf{is\text{-}zero}(a) \uplus \mathsf{is\text{-}str\text{-}suc}(a) \uplus \mathsf{is\text{-}limit}(a)$ is contractible.

▶ **Theorem 11.** Cnf *and* Brw *have classification.* Ord *having classification would imply the law of excluded middle.* ◀

Classifiability corresponds to a case distinction, but the useful principle from classical ordinal theory is the related induction principle:

▶ **Definition 12** (classifiability induction)**.** *We say that $(A, <, \leq)$ satisfies the principle of* classifiability induction *if the following holds: For every family $P : A \to \mathsf{hProp}$ such that*

$$\begin{aligned}
&\mathsf{is\text{-}zero}(a) \to P(a)\\
&(a \mathrel{\mathsf{is\text{-}str\text{-}suc\text{-}of}} b) \to P(b) \to P(a)\\
&(a \mathrel{\mathsf{is\text{-}lim\text{-}of}} f) \to (\forall i.P(f_i)) \to P(a),
\end{aligned}$$

we have $\forall a.P(a)$.

Note that classifiability induction does *not* ask for successors or limits to be computable. Using Lemma 10, we get that classifiability induction implies classification. For the reverse, we need a further assumption:

▶ **Theorem 13.** *Assume $(A, <, \leq)$ has classification and satisfies the principle of transfinite induction. Then $(A, <, \leq)$ satisfies the principle of classifiability induction.* ◀

It is also standard in classical set theory that classifiability induction implies transfinite induction: showing P by transfinite induction corresponds to showing $\forall x < a.P(x)$ by classifiability induction. In our setting, this would require strong additional assumptions, including the assumption that $(x \leq a)$ is equivalent to $(x < a) \uplus (x = a)$, i.e. that $\leq$ is the reflexive closure of $<$. The standard proof works with several strong assumptions of this form, but we do not consider this interesting or useful, and concentrate on the results which work for the weaker assumptions that are satisfied for Brw and Ord (see Section 4.3.1).

▶ **Theorem 14.** Cnf *and* Brw *satisfy classifiability induction, while* Ord *satisfying it again implies excluded middle.* ◀

4.4 Arithmetic

Using the predicates $\mathsf{is\text{-}zero}(a)$, $a \mathrel{\mathsf{is\text{-}suc\text{-}of}} b$, and $a \mathrel{\mathsf{is\text{-}sup\text{-}of}} f$, we can define what it means for $(A, <, \leq)$ to have the standard arithmetic operations. We still work under the assumptions declared in Section 4.3.1 — in particular, we do not assume that e.g. limits can be calculated, which is important to make the theory applicable to Cnf.

▶ **Definition 15** (having addition). *We say that* $(A,<,\le)$ has addition *if there is a function* $+ : A \to A \to A$ *which satisfies the following properties:*

$$\begin{aligned}
&\mathsf{is\text{-}zero}(a) \to c + a = c\\
&a \;\mathsf{is\text{-}suc\text{-}of}\; b \to d \;\mathsf{is\text{-}suc\text{-}of}\; (c+b) \to c + a = d\\
&a \;\mathsf{is\text{-}lim\text{-}of}\; f \to b \;\mathsf{is\text{-}sup\text{-}of}\; (\lambda i.c + f_i) \to c + a = b \qquad (3)
\end{aligned}$$

We say that A has unique addition *if there is exactly one function* $+$ *with these properties.*

Note that (3) makes an assumption only for (strictly) *increasing* sequences f; this suffices to define a well-behaved notion of addition, and it is not necessary to include a similar requirement for arbitrary sequences. Since $(\lambda i.c + f_i)$ is a priori not necessarily increasing, the middle term of (3) has to talk about the supremum, not the limit.

Completely analogously to addition, we can formulate multiplication and exponentation, again without assuming that successors or limits can be calculated:

▶ **Definition 16** (having multiplication). *Assuming that* A *has addition, we say that it* has multiplication *if we have a function* $\cdot : A \to A \to A$ *that satisfies the following properties:*

$$\begin{aligned}
&\mathsf{is\text{-}zero}(a) \to c \cdot a = a\\
&a \;\mathsf{is\text{-}suc\text{-}of}\; b \to c \cdot a = c \cdot b + c\\
&a \;\mathsf{is\text{-}lim\text{-}of}\; f \to b \;\mathsf{is\text{-}sup\text{-}of}\; (\lambda i.c \cdot f_i) \to c \cdot a = b
\end{aligned}$$

A has unique multiplication *if it has unique addition and there is exactly one function* $\cdot$ *with the above properties.*

▶ **Definition 17** (having exponentation). *Assume* A *has addition and multiplication. We say that* A has exponentation with base c *if we have a function* $\mathsf{exp}(c,-) : A \to A$ *that satisfies the following properties:*

$$\begin{aligned}
&\mathsf{is\text{-}zero}(b) \to a \;\mathsf{is\text{-}suc\text{-}of}\; b \to \mathsf{exp}(c,b) = a\\
&a \;\mathsf{is\text{-}suc\text{-}of}\; b \to \mathsf{exp}(c,a) = \mathsf{exp}(c,b) \cdot c\\
&a \;\mathsf{is\text{-}lim\text{-}of}\; f \to \neg\mathsf{is\text{-}zero}(c) \to b \;\mathsf{is\text{-}sup\text{-}of}\; (\mathsf{exp}(c,f_i)) \to \mathsf{exp}(c,a) = b\\
&a \;\mathsf{is\text{-}lim\text{-}of}\; f \to \mathsf{is\text{-}zero}(c) \to \mathsf{exp}(c,a) = c
\end{aligned}$$

A has unique exponentation with base c *if it has unique addition and multiplication, and if* $\mathsf{exp}(c,-)$ *is unique.*

▶ **Theorem 18.** Cnf *has addition, multiplication, and exponentiation with base* ω *(all unique),* Brw *has addition, multiplication and exponentiation with every base (all unique), and* Ord *has addition and multiplication.* ◀⚙

For Cnf, arithmetic is defined by pattern matching on the trees. Addition[3] is given as

$$\begin{aligned}
0 + b &:\equiv b\\
a + 0 &:\equiv a\\
(\omega^a \boldsymbol{+} c) + (\omega^b \boldsymbol{+} d) &:\equiv \begin{cases} \omega^b \boldsymbol{+} d & \text{if } a < b\\ \omega^a \boldsymbol{+} (c + (\omega^b \boldsymbol{+} d)) & \text{otherwise,}\end{cases}
\end{aligned}$$

[3] Caveat: $\boldsymbol{+}$ is a notation for the tree constructor, while $+$ is an operation that we define. We use parenthesis so that all operations can be read with the usual operator precedence.

multiplication as

$$\begin{aligned}
0 \cdot b &:\equiv 0 \\
a \cdot 0 &:\equiv 0 \\
a \cdot (\omega^0 + d) &:\equiv a + a \cdot d \\
(\omega^a + c) \cdot (\omega^b + d) &:\equiv (\omega^{a+b} + 0) + (\omega^a + c) \cdot d \quad \text{if } b \neq 0,
\end{aligned}$$

and exponentiation with base ω by $\omega^a :\equiv \omega^a + 0$. These definitions are standard. Novel is our proof of correctness in the sense of Definitions 15–17, which we achieve by defining the inverse operations of subtraction and division.

Arithmetic on Brw is defined by recursion on the second argument, following the clauses of the specifications in Definitions 15–17. Since the constructor limit only accepts an increasing sequence, it is necessary to prove mutually with the definition that the operations are monotone and preserve increasing sequences. However, the case $c = 0$ needs to be treated separately since neither pointwise multiplication nor exponentiation with 0 preserves increasingness. This makes it crucial to use classification (Theorem 11) and, in particular, that it is decidable whether $c : \mathsf{Brw}$ is zero.

Addition on Ord is given by disjoint union $A \uplus B$ (with $\mathsf{inl}(a) \prec_{A \uplus B} \mathsf{inr}(b)$), and multiplication by Cartesian product $A \times B$ with the reverse lexicographical order. We expect that exponentation cannot be defined constructively: the "obvious" definition via function spaces gives a wellfounded order assuming the law of excluded middle [27], but it seems unlikely that it can be avoided.

5 Interpretations Between the Notions

In this section, we show how our three notions of ordinals can be connected via structure preserving embeddings.

5.1 From Cantor Normal Forms to Brouwer Trees

The arithmetic operations of Brw allow the construction of a function $\mathsf{CtoB} : \mathsf{Cnf} \to \mathsf{Brw}$ in a canonical way. We define $\mathsf{CtoB} : \mathsf{Cnf} \to \mathsf{Brw}$ by:

$$\begin{aligned}
\mathsf{CtoB}(0) &:\equiv \mathsf{zero} \\
\mathsf{CtoB}(\omega^a + b) &:\equiv \omega^{\mathsf{CtoB}(a)} + \mathsf{CtoB}(b)
\end{aligned}$$

▶ **Theorem 19.** *The function* CtoB *preserves and reflects* $<$ *and* $\leq$*, i.e.,* $a < b \leftrightarrow \mathsf{CtoB}(a) < \mathsf{CtoB}(b)$*, and* $a \leq b \leftrightarrow \mathsf{CtoB}(a) \leq \mathsf{CtoB}(b)$*.* ◀

To show that CtoB preserves $<$, we first prove that Brouwer trees of the form ω^x are additive principal: if $a < \omega^x$ then $a + \omega^x = \omega^x$ — a property not true for the "naive" version of Brouwer trees without path constructors. By reflecting $\leq$ and antisymmetry, we have:

▶ **Corollary 20.** *The function* CtoB *is injective.* ◀

We note that CtoB also preserves all arithmetic operations on Cnf. For multiplication, this relies on $\iota(n) \cdot \omega^x = \omega^x$ for Brw, where $\iota : \mathbb{N} \to \mathsf{Brw}$ embeds the natural numbers as Brouwer trees, and $\omega :\equiv \mathsf{limit}\,\iota$ — see our formalisation for details.

▶ **Theorem 21.** CtoB *commutes with addition, multiplication, and exponentiation with base* ω*.* ◀

Lastly, as expected, Brouwer trees define bigger ordinals than Cantor normal forms: when embedded into Brw, all Cantor normal forms are below ε_0, the limit of the increasing sequence ω, ω^ω, ω^{ω^ω}, ...

▶ **Theorem 22.** *For all* $a : \mathsf{Cnf}$*, we have* $\mathsf{CtoB}(a) < \mathsf{limit}\,(\lambda k.\omega \uparrow\uparrow k)$*, where* $\omega \uparrow\uparrow 0 :\equiv \omega$ *and* $\omega \uparrow\uparrow (k+1) :\equiv \omega^{\omega \uparrow\uparrow k}$. ◀

5.2 From Brouwer Trees to Extensional Wellfounded Orders

As Brw comes with an order that is extensional, wellfounded, and transitive, it can itself be seen as an element of Ord. Every "subtype" of Brw (constructed by restricting to trees smaller than a given tree) inherits this property, giving a canonical function from Brouwer trees to extensional, wellfounded orders. We define

$$\mathsf{BtoO}(a) = \Sigma(y : \mathsf{Brw}).(y < a).$$

with order relation $(y, p) \prec (y', p')$ if $y < y'$. This extends to a function $\mathsf{BtoO} : \mathsf{Brw} \to \mathsf{Ord}$. The first projection gives a simulation $\mathsf{BtoO}(a) \leq \mathsf{Brw}$. Using extensionality of Brw, this implies that BtoO is an embedding from Brw into Ord. Using that $<$ on Brw is propositional, and that carriers of orders are sets, it is also not hard to see that BtoO is order-preserving:

▶ **Lemma 23.** *The function* $\mathsf{BtoO} : \mathsf{Brw} \to \mathsf{Ord}$ *is injective, and preserves* $<$ *and* $\leq$. ◀

A natural question is whether the above result can be strengthened further, i.e. whether BtoO is a simulation. Using LEM to find a minimal simulation witness, this is possible:

▶ **Theorem 24.** *Under the assumption of the law of excluded middle, the function* $\mathsf{BtoO} : \mathsf{Brw} \to \mathsf{Ord}$ *is a simulation.* ◀

We do not know whether the reverse of Theorem 24 is provable, but from the assumption that BtoO is a simulation, we can derive another constructive taboo:

▶ **Theorem 25.** *If the map* $\mathsf{BtoO} : \mathsf{Brw} \to \mathsf{Ord}$ *is a simulation, then WLPO holds.* ◀

We trivially have $\mathsf{BtoO}(\mathsf{zero}) = \mathbf{0}$. One can further prove that BtoO commutes with limits, i.e. $\mathsf{BtoO}(\mathsf{limit}(f)) = \lim(\mathsf{BtoO} \circ f)$. However, BtoO does *not* commute with successors; it is easy to see that $\mathsf{BtoO}\, x \uplus \mathbf{1} \leq \mathsf{BtoO}(\mathsf{succ}\, x)$, but the other direction implies WLPO. This also means that BtoO does not preserve the arithmetic operations but "over-approximates" them, i.e. we have $\mathsf{BtoO}(x + y) \geq \mathsf{BtoO}\, x \uplus \mathsf{BtoO}\, y$ and $\mathsf{BtoO}(x \cdot y) \geq \mathsf{BtoO}\, x \times \mathsf{BtoO}\, y$.

6 Conclusions and Future Directions

We have demonstrated that three very different implementations of ordinal numbers, namely Cantor normal forms (Cnf), Brouwer ordinal trees (Brw), and extensional wellfounded orders (Ord), can be studied in a single abstract setting in the context of homotopy type theory. We hope that our development may shed light on other constructive or formalised approaches to ordinals also in other settings [31, 7, 6, 34].

Cantor normal forms are a formulation where most properties are decidable, while the opposite is the case for extensional wellfounded orders. Brouwer ordinal trees sit in the middle, with some of its properties being decidable. This aspect is not discussed in full in this paper; we only have included Theorem 14. It is easy to see that, for $x : \mathsf{Brw}$, it is decidable whether x is finite; in other words, the predicate $(\omega \leq \_) : \mathsf{Brw} \to \mathsf{hProp}$ is decidable, while $(\omega < \_)$ is decidable if and only if WLPO holds.

If x is finite, then the predicates $(x = \_)$, $(x \leq \_)$, and $(x < \_)$ are also decidable. We have a further proof that, if $c : \mathsf{Cnf}$ is smaller than ω^2, then the families $(\mathsf{CtoB}\, c \leq \_)$ and $(\mathsf{CtoB}\, c < \_)$ are *semidecidable*, where semidecidability can be defined using the *Sierpinski space* [3, 13, 42].

Thus, each of the canonical maps $\mathsf{CtoB} : \mathsf{Cnf} \to \mathsf{Brw}$ and $\mathsf{BtoO} : \mathsf{Brw} \to \mathsf{Ord}$ embeds the "more decidable" formulation of ordinals into the "less decidable" one. Naturally, they both also include a "smaller" type of ordinals into a "larger" one: While every element of Cnf represents an ordinal below ϵ_0, Brw can go much further. It would be interesting to consider more powerful ordinal notation systems such as those based on the Veblen functions [41, 35] or collapsing functions[4, 9], and see how these compare to Brouwer trees. Another avenue for potentially extending Cantor normal forms would be using *superleaves* [19]; we do not know how such a "bigger" version of Cnf would compare to Brw.

Since Brw can be viewed as an element of Ord, the latter can clearly reach larger ordinals than the former. This is of course not surprising; the Burali-Forti argument [5, 11] shows that lower universes cannot reach the same ordinals as higher universes. Another obstruction for Brw to reach the full power of Ord is the fact that Brw only includes limits of $\mathbb{N}$-indexed sequences. To overcome this problem, one can similarly construct higher number classes as quotient inductive-inductive types, e.g. a type Brw_3 closed under limits of Brw-indexed sequences, and then more generally types Brw_{n+1} closed under limits of Brw_n-indexed sequences, and so on.

Finally, there are interesting connections between the ordinals we can represent and the proof-theoretic strength of the ambient type theory: each proof of wellfoundedness for a system of ordinals is also a lower bound for the strength of the type theory it is constructed in. It is well known that definitional principles such as simultaneous inductive-recursive definitions [22] and higher inductive types [30] can increase the proof-theoretical strength, and so, we hope that they can also be used to faithfully represent even larger ordinals.

References

1 Peter Aczel. An introduction to inductive definitions. In *Studies in Logic and the Foundations of Mathematics*, volume 90, pages 739–782. Elsevier, 1977.

2 Thorsten Altenkirch, Paolo Capriotti, Gabe Dijkstra, Nicolai Kraus, and Fredrik Nordvall Forsberg. Quotient inductive-inductive types. In Christel Baier and Ugo Dal Lago, editors, *FoSSaCS '18*, pages 293–310. Springer, 2018.

3 Thorsten Altenkirch, Nils Anders Danielsson, and Nicolai Kraus. Partiality, revisited. In Javier Esparza and Andrzej S. Murawski, editors, *FoSSaCS '17*, pages 534–549. Springer, 2017.

4 Heinz Bachmann. Die Normalfunktionen und das Problem der ausgezeichneten Folgen von Ordnungszahlen. *Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich*, 2:115–147, 1950.

5 Marc Bezem, Thierry Coquand, Peter Dybjer, and Martín Escardó. The Burali-Forti argument in HoTT/UF in Agda notation, 2020. Available at `https://www.cs.bham.ac.uk/~mhe/TypeTopology/BuraliForti.html`.

6 Jasmin Christian Blanchette, Mathias Fleury, and Dmitriy Traytel. Nested multisets, hereditary multisets, and syntactic ordinals in Isabelle/HOL. In Dale Miller, editor, *FSCD '17*, volume 84 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:18, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

7 Jasmin Christian Blanchette, Andrei Popescu, and Dmitriy Traytel. Cardinals in Isabelle/HOL. In G. Klein and R. Gamboa, editors, *ITP '14*, volume 8558 of *Lecture Notes in Computer Science*, pages 111–127, 2014.

8 L. E. J. Brouwer. Zur begründung der intuitionistische mathematik III. *Mathematische Annalen*, 96:451–488, 1926.

9 Wilfried Buchholz. A new system of proof-theoretic ordinal functions. *Annals of Pure and Applied Logic*, 32:195–207, 1986.

10 Wilfried Buchholz. Notation systems for infinitary derivations. *Archive for Mathematical Logic*, 30:227–296, 1991.

11 Cesare Burali-Forti. Una questione sui numeri transfiniti. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, 11(1):154–164, 1897.

12 Pierre Castéran and Evelyne Contejean. On ordinal notations. Available at `http://coq.inria.fr/V8.2pl1/contribs/Cantor.html`, 2006.

13 James Chapman, Tarmo Uustalu, and Niccolò Veltri. Quotienting the delay monad by weak bisimilarity. *Mathematical Structures in Computer Science*, 29(1):67–92, 2019.

14 Alonzo Church. The constructive second number class. *Bulletin of the American Mathematical Society*, 44:224–232, 1938.

15 The Coq Development Team. *The Coq proof assistant reference manual*, 2021.

16 Thierry Coquand, Peter Hancock, and Anton Setzer. Ordinals in type theory. Invited talk at Computer Science Logic (CSL), `http://www.cse.chalmers.se/~coquand/ordinal.ps`, 1997.

17 Thierry Coquand, Simon Huber, and Anders Mörtberg. On higher inductive types in cubical type theory. *LICS '18*, 2018.

18 Leonardo Mendonça de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. The Lean theorem prover (system description). In Amy P. Felty and Aart Middeldorp, editors, *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*, volume 9195 of *Lecture Notes in Computer Science*, pages 378–388. Springer, 2015.

19 Nachum Dershowitz. Trees, ordinals and termination. In M. C. Gaudel and J. P. Jouannaud, editors, *TAPSOFT '93*, pages 243–250. Springer, 1993.

20 Nachum Dershowitz and Zohar Manna. Proving termination with multiset orderings. *Communications of the ACM*, 22(8):465–476, 1979.

21 Gabe Dijkstra. *Quotient inductive-inductive definitions.* PhD thesis, University of Nottingham, 2017.

22 Peter Dybjer and Anton Setzer. Induction-recursion and initial algebras. *Annals of Pure and Applied Logic*, 124(1):1–47, 2003.

23 Martín Escardó. Compact ordinals, discrete ordinals and their relationships, Since 2010–2021. Available at `https://www.cs.bham.ac.uk/~mhe/TypeTopology/Ordinals.html`.

24 Robert W. Floyd. Assigning meanings to programs. In J.T. Schwartz, editor, *Symposium on Applied Mathematics*, volume 19, pages 19–32, 1967.

25 José Grimm. Implementation of three types of ordinals in Coq. Technical Report RR-8407, INRIA, 2013. Available at `https://hal.inria.fr/hal-00911710`.

26 Peter Hancock. *Ordinals and Interactive Programs.* PhD thesis, University of Edinburgh, 2000.

27 W Charles Holland, Salma Kuhlmann, and Stephen H McCleary. Lexicographic exponentiation of chains. *Journal of Symbolic Logic*, pages 389–409, 2005.

28 S. C. Kleene. On notation for ordinal numbers. *The Journal of Symbolic Logic*, 3(4):150–155, 1938.

29 Daniel Licata and Michael Shulman. Calculating the fundamental group of the circle in homotopy type theory. In *LICS '13*, pages 223–232, 2013.

30 Peter Lefanu Lumsdaine and Michael Shulman. Semantics of higher inductive types. *Mathematical Proceedings of the Cambridge Philosophical Society*, 169(1):159–208, 2020.

31 Panagiotis Manolios and Daron Vroon. Ordinal arithmetic: algorithms and mechanization. *Journal of Automated Reasoning*, 34(4):387–423, 2005.

32 Fredrik Nordvall Forsberg, Chuangjie Xu, and Neil Ghani. Three equivalent ordinal notation systems in cubical Agda. In *CPP '20*, pages 172–185. ACM, 2020.

33 Ulf Norell. *Towards a practical programming language based on dependent type theory.* PhD thesis, Chalmers University of Technology, 2007.

34 Peter H. Schmitt. A mechanizable first-order theory of ordinals. In R. Schmidt and C. Nalon, editors, *TABLEAUX '17*, volume 10501 of *Lecture Notes in Computer Science*, pages 331–346, 2017.

35 Kurt Schütte. Kennzeichnung von Ordinalzahlen durch rekursiv erklärte Funktionen. *Mathematische Annalen*, 127:15–32, 1954.

36 Kurt Schütte. *Proof Theory.* Springer, 1977.

37 Michael Shulman. The surreals contain the plump ordinals. Blog post at `https://homotopytypetheory.org/2014/02/22/surreals-plump-ordinals/`, 2014.

38 Gaisi Takeuti. *Proof Theory.* North-Holland, 2 edition, 1987.

39 Paul Taylor. Intuitionistic sets and ordinals. *Journal of Symbolic Logic*, 61(3):705–744, 1996.

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

41 Oswald Veblen. Continuous increasing functions of finite and transfinite ordinals. *Transactions of the American Mathematical Society*, 9(3):280–292, 1908.

42 Niccolò Veltri. *A type-theoretical study of nontermination.* PhD thesis, Tallinn University of Technology, 2017.

43 Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. Cubical Agda: a dependently typed programming language with univalence and higher inductive types. In *ICFP '19*, volume 3, pages 87:1–87:29. ACM, 2019.

Appendix

In addition to our Agda formalisation, we include paper proofs of the results listed in the paper, organised as follows:

- Appendix A contains arguments for the statements that are formulated in our abstract framework;
- Appendix B proves the results for Cantor normal forms;
- Appendix C proves the theorems about Brouwer trees;
- Appendix D gives the arguments for extensional wellfounded orders;
- and Appendix E contains proofs for Section 5.

A General proofs

▶ **Lemma 3** (Transfinite Induction). *Let $<$ be wellfounded and $P : A \to \mathcal{U}$ be a type family such that $\forall a.(\forall b < a.P(b)) \to P(a)$. Then, it follows that $\forall a.P(a)$.*

Proof. This is a special case of the induction principle that one gets from the inductive definition of the accessibility family. ◀

▶ **Lemma 7.** *Let $s : A \to A$ be given. The function s calculates successors if and only if $\forall bx.(b < x) \leftrightarrow (s\,b \leq x)$.*

Proof. Assume that s calculates successors; we have to show $(b < x) \leftrightarrow (s\,b \leq x)$. The part $\to$ is clear. Further, since $s\,b$ is the successor of b, we have $b < s\,b$. Together with $(<) \subseteq (\leq)$ and $(< \circ \leq) \subseteq (<)$, this shows the part $\leftarrow$.

Now assume $(b < x) \leftrightarrow (s\,b \leq x)$. By reflexivity of $\leq$, the case $x \equiv s\,b$ shows $b < s\,b$, and the direction $\to$ directly gives the second part of "s calculates successors". ◀

▶ **Lemma 10.** *Any $a : A$ can be at most one out of {zero, strong successor, limit}, and in a unique way. In other words, the type $\textsf{is-zero}(a) \uplus \textsf{is-str-suc}(a) \uplus \textsf{is-limit}(a)$ is a proposition.*

Proof. First part: Each of the three summands is individually a proposition. This is clear for $\mathsf{is\text{-}zero}(a)$ and $\mathsf{is\text{-}limit}(a)$. Assume that a is the strong successor of both b and b'; then we have $b \leq b'$ and $b' \leq b$, implying $b = b'$ by antisymmetry.

Second part: We now only have to check that any two summands exclude each other. Since the goal is a proposition, we can assume that we are given b and f in the successor and limit case. Assume that a is zero and the successor of b. This implies $b < a \leq b$ and thus $b < b$, contradicting irreflexivity. If a is zero and the limit of f, the same argument (with b replaced by f_0) applies. Finally, assume that a is the strong successor of b and the limit of f. These assumptions show that b is an upper bound of f, thus we get $a \leq b$. Together with $b < a$, this gives the contradiction $b < b$ as above. ◀

▶ **Theorem 13.** *Assume $(A, <, \leq)$ has classification and satisfies the principle of transfinite induction. Then $(A, <, \leq)$ satisfies the principle of classifiability induction: for every family $P : A \to \mathsf{hProp}$ such that*

$$\mathsf{is\text{-}zero}(a) \to P(a) \tag{4}$$

$$(a \ \mathsf{is\text{-}str\text{-}suc\text{-}of} \ b) \to P(b) \to P(a) \tag{5}$$

$$(a \ \mathsf{is\text{-}lim\text{-}of} \ f) \to (\forall i.P(f_i)) \to P(a), \tag{6}$$

we have $\forall a.P(a)$.

Proof. By transfinite induction, it suffices to show

$$(\forall b < a.P(b)) \to P(a) \tag{7}$$

for some fixed a. By classification, we can consider three cases. If $\mathsf{is\text{-}zero}(a)$, then (4) gives us $P(a)$, which shows (7) for that case. If a is the strong successor of b, we use that the predecessor b is one of the elements that the assumption of (7) quantifies over; therefore, this is implied by (5). Similarly, if $\mathsf{is\text{-}lim}(a)$, the assumption of (7) gives $\forall i.P(f_i)$, thus (6) gives $P(a)$. ◀

▶ **Lemma 26.** *If A satisfies the principle of classifiability induction then it also has classification.*

Proof. By Lemma 10, we can use the type

$$\mathsf{is\text{-}zero}(a) \uplus \mathsf{is\text{-}str\text{-}suc}(a) \uplus \mathsf{is\text{-}limit}(a) \tag{8}$$

as the motive of a straightforward classifiability induction. ◀

B Proofs for Cantor Normal Forms

We firstly prove the theorems about properties of Cnf and its relations.

▶ **Theorem 1** (for Cantor normal forms). *Cnf is a set and the relations $<$ and $\leq$ are valued in propositions. In addition, $<$ and $\leq$ are transitive, $<$ is irreflexive, and $\leq$ is reflexive and antisymmetric. $<$ is trichotomous and $\leq$ connex.*

Proof. Most properties are proved by induction on the arguments. We prove the trichotomy property as an example. It is trivial when either argument is zero. Given $\omega^a + b$ and $\omega^c + d$, by induction hypothesis we have $a < c \uplus a = c \uplus c < a$ correspondingly. For the first and last cases, we have $\omega^a + b < \omega^c + d$ and $\omega^c + d < \omega^a + b$. For the middle case, the induction hypothesis on b and d gives the desired result. ◀

▶ **Theorem 2** (for Cantor normal forms). *The relations $<$ and $\leq$ on* Cnf *are extensional.*

Proof. Extensionality is trivial for an irreflexive and trichotomous relation, which covers $<$, and is also trivial for a reflexive and antisymmetric relation, which covers $\leq$. ◀

▶ **Theorem 4** (for Cantor normal forms). *The relation $<$ on* Cnf *is wellfounded.*

Proof. See the proof of [32, Theorem 5.1]. ◀

▶ **Theorem 5** (for Cantor normal forms). Cnf *and its relations satisfy:*
(A1) *$<$ is transitive and irreflexive;*
(A2) *$\leq$ is reflexive, transitive, and antisymmetric;*
(A3) *we have $(<) \subseteq (\leq)$ and $(< \circ \leq) \subseteq (<)$.*

Proof. By induction on arguments as in the proof of Theorem 1. ◀

▶ **Theorem 6** (for Cantor normal forms). Cnf *has a zero.*

Proof. The element $0 : \mathsf{Cnf}$ is a zero. ◀

▶ **Theorem 8** (for Cantor normal forms). Cnf *has strong successors, and it is both $<$- and $\leq$-monotone.*

Proof. It is trivial to show that $a + 1$ is a strong successor of a. The monotonicity properties of $\_ + 1$ is proved by induction on the argument. ◀

▶ **Theorem 9** (for Cantor normal forms). Cnf *does not have suprema or limits. Assuming the law of exclude middle,* Cnf *has suprema (and thus limits) of arbitrary* bounded *sequences. If* Cnf *has limits of bounded increasing sequences, then the weak limited principle of omniscience (WLPO) is derivable.*

Proof. To show that Cnf does not have suprema or limits, we use the sequence $\omega \uparrow\uparrow$ from Theorem 22 as a counterexample. Recall $\omega \uparrow\uparrow 0 :\equiv \omega$ and $\omega \uparrow\uparrow (k + 1) :\equiv \omega^{\omega \uparrow\uparrow k}$. If it has a limit, say $x : \mathsf{Cnf}$, then any CNF a is strictly smaller than x, including x itself. But this contradicts to irreflexivity.

Assume LEM. Given a bounded sequence f with bound b, consider the subset $P : \mathsf{Cnf} \to \mathsf{hProp}$ of all Cantor normal forms that are upper bounds of f. This subset contains at least b and is thus non-empty. Since $<$ on Cnf is extensional and wellfounded, [40, Thm 10.4.3] implies that P has a least element, and this least upper bound is a supremum of f.

The last part uses arithmetic for Cantor normal forms. If Cnf has limits of bounded increasing sequences, assume $s : \mathbb{N} \to \mathbf{2}$ is a given sequence. Define $f : \mathbb{N} \to \mathsf{Cnf}$ by $f_0 :\equiv 0$ and

$$f_{n+1} :\equiv \begin{cases} f_n + 1 & \text{if } s_n = \mathsf{ff} \\ \mathsf{max}(f_n, \omega) + 1 & \text{if } s_n = \mathsf{tt}. \end{cases} \tag{9}$$

The increasing sequence is bounded by $\omega + \omega$. We can check whether its limit is ω, in which case the original sequence s is constantly ff, otherwise it is not. ◀

Even though we cannot compute limits of arbitrary sequences in Cnf, if a x is not zero or a successor, then we can always find a *fundamental sequence* whose limit is x — this is enough to establish the following:

▶ **Lemma 27.** *If a CNF is neither zero nor a successor, then it is a limit.*

Proof. If a CNF x is neither zero nor a successor, then $x = \omega^a + 0$ with $a > 0$ or $x = \omega^a + b$ where $b > 0$ is not a successor. There are three possible cases, for each of which we construct an increasing sequence $s : \mathbb{N} \to \mathsf{Cnf}$ whose limit is x:

(i) If $x = \omega^a + 0$ and $a = c + 1$, we define $s_i :\equiv (\omega^c + 0) \cdot \eta i$ where $\eta : \mathbb{N} \to \mathsf{Cnf}$ embeds natural numbers to CNFs.

(ii) If $x = \omega^a + 0$ and a is not a successor, the induction hypothesis on a gives a sequence r, and then we define $s_i :\equiv \omega^{r_i} + 0$.

(iii) If $x = \omega^a + b$ and $b > 0$ is not a successor, the induction hypothesis on b gives a sequence r, and then we define $s_i :\equiv \omega^a + r_i$. ◀

▶ **Theorem 11** (for Cantor normal forms). Cnf *has classification.*

Proof. Since Cnf has decidable equality, being zero and being a successor are both decidable. Then Lemma 27 gives the desired result. ◀

▶ **Theorem 14** (for Cantor normal forms). Cnf *satisfies classifiability induction.*

Proof. By Theorems 4, 11, and 13. ◀

To show that the operations $(+)$ and $(\cdot)$ correctly implement addition and multiplication in the sense of Definitions 15 and 16, we construct their inverse operations *subtraction* and *division*:

▶ **Lemma 28** (Subtraction and division of CNFs). *For all CNFs a, b,*

1. *if $a \leq b$, then there is a CNF c such that $a + c = d$;*
2. *if $b > 0$, then there are CNFs c and d such that $a = b \cdot c + d$ and $d < b$.*

Proof. For 1 we define subtraction as follows:

$$\begin{aligned} 0 - b &:\equiv 0 \\ a - 0 &:\equiv a \\ (\omega^a + c) - (\omega^b + d) &:\equiv \begin{cases} 0 & \text{if } a < b \\ c - d & \text{if } a = b \\ \omega^a + c & \text{if } a > b. \end{cases} \end{aligned}$$

The proof of 2 consists of the following cases:

- If $a < b$, then we take $c :\equiv 0$ and $d :\equiv a$.
- If $a = b$, then we take $c :\equiv 1$ and $d :\equiv 0$.
- If $a > b$, then there two possibilities:

(i) $a = \omega^u + u'$ and $b = \omega^v + v'$ with $u > v$. By the induction hypothesis on u' and b, we have c' and d such that $u' = b \cdot c' + d$ and $d < b$. We take $c :\equiv \omega^{(u-v)} + c'$ and then have $a = \omega^u + u' = b \cdot \omega^{(a-b)} + b \cdot c' + d = b \cdot c + d$.

(ii) $a = \omega^u + u'$ and $b = \omega^u + v'$ with $u' > v'$. By the induction hypothesis on $u' - v'$ and b, we have c' and d such that $u' - v' = b \cdot c' + d$ and $d < b$. We take $c :\equiv c' + 1$ and then have $a = \omega^u + u' = \omega^u + v' + (u' - v') = b + b \cdot c' + d = b \cdot c + d$.

The above defines the (Euclidean) division of CNFs. ◀

▶ **Theorem 18** (for Cantor normal forms). Cnf *has addition, multiplication, and exponentiation with base ω, and all operations are unique.*

Proof. As an example, we verify that $(+)$ implements addition, i.e., it satisfies

1. $\mathsf{is\text{-}zero}(a) \to c + a = c$,

2. a is-suc-of $b \to d$ is-suc-of $(c+b) \to c+a=d$, and
3. a is-lim-of $f \to b$ is-sup-of $(\lambda i.c+f_i) \to c+a=b$.

For 1, it suffices to show that 0 is the additive identity, which is easy. For 2, it suffices to show $c+(b+1)=(c+b)+1$ which holds by associativity of $(+)$. For 3, it suffices to show that $c+a$ is the limit of $\lambda i.c+f_i$. Firstly, we have $c+f_i \leq c+a$ for all i because a is the limit of f and $(+)$ is monotone on the right argument. It remains to show that $c+a$ is at least as large as any x with $c+f_i \leq x$ for all i. Given such x, we have $f_i \leq x-c$ for all i. Because a is the limit of f, we have $a \leq x-c$. Therefore, we have $c+a \leq c+(x-c)=x$ where the equation holds due to Lemma 28.1. ◀

C Proofs about Brouwer Trees

▶ **Lemma 29.** *The constructors of* Brw *are distinguishable, i.e. one can construct proofs of* zero $\neq$ succ x*,* zero $\neq$ limit f*, and* succ $x \neq$ limit f*.*

Proof. Point constructors are not always distinct in the presence of path constructors. Nevertheless, this is fairly simple in our case, as the path constructor bisim only equates limits, and the standard strategy of simply defining distinguishing families such as isZero : Brw → hProp works. Setting

$$\begin{aligned}
\mathsf{isZero}\ \mathsf{zero} &:\equiv \mathbf{1}\\
\mathsf{isZero}\ (\mathsf{succ}\,x) &:\equiv \mathbf{0}\\
\mathsf{isZero}\ (\mathsf{limit}\,f) &:\equiv \mathbf{0}
\end{aligned}$$

means the proof obligations for the path constructors (bisim and the truncation constructor) are trivial. Now if zero $=$ succ x, since isZero zero is inhabited, isZero (succ x) $\equiv \mathbf{0}$ must be as well — a contradiction, which shows $\neg$(zero $=$ succ x).

In the same way, one shows the other parts. For the details, we refer to our formalisation. ◀

Characterisation of $\leq$

In order to prove some of the non-trivial properties of Brw, it is necessary to characterise the relation $\leq$. We use a strategy corresponding to the *encode-decode method* [29] and define the type family

$$\mathsf{Code} : \mathsf{Brw} \to \mathsf{Brw} \to \mathsf{hProp}$$

mutually with proofs of its *correctness* properties

$$\begin{aligned}
\mathsf{toCode} &: x \leq y \to \mathsf{Code}\,x\,y\\
\mathsf{fromCode} &: \mathsf{Code}\,x\,y \to x \leq y,
\end{aligned}$$

for every $x, y : \mathsf{Brw}$, with the goal of providing a concrete description of $x \le y$. On the point constructors, the definition works as follows:

$$
\begin{array}{lllll}
\mathsf{Code} & \mathsf{zero} & \_ & :\equiv & \mathbf{1} \\
\mathsf{Code} & (\mathsf{succ}\, x) & \mathsf{zero} & :\equiv & \mathbf{0} \\
\mathsf{Code} & (\mathsf{succ}\, x) & (\mathsf{succ}\, y) & :\equiv & \mathsf{Code}\; x\; y \quad (10) \\
\mathsf{Code} & (\mathsf{succ}\, x) & (\mathsf{limit}\, f) & :\equiv & \exists n. \mathsf{Code}\, (\mathsf{succ}\, x)\, (f\, n) \quad (11) \\
\mathsf{Code} & (\mathsf{limit}\, f) & \mathsf{zero} & :\equiv & \mathbf{0} \\
\mathsf{Code} & (\mathsf{limit}\, f) & (\mathsf{succ}\, y) & :\equiv & \forall k. \mathsf{Code}\, (f\, k)\, (\mathsf{succ}\, y) \quad (12) \\
\mathsf{Code} & (\mathsf{limit}\, f) & (\mathsf{limit}\, g) & :\equiv & \forall k. \exists n. \mathsf{Code}\, (f\, k)\, (g\, n) \quad (13)
\end{array}
$$

As explained in the paper, the path constructor bisim proves to be challenging. We refer to our Agda formalisation for the details.

▶ **Lemma 30.** *For $x, y : \mathsf{Brw}$, the type $\mathsf{Code}\, x\, y$ is equivalent to $x \le y$.*

Proof. Both types are propositions, and maps in both directions are given by toCode and fromCode. ◀

Code allows us to easily derive various useful auxiliary lemmas, for example the following four:

▶ **Lemma 31.** *For $x, y : \mathsf{Brw}$, we have $x \le y \leftrightarrow \mathsf{succ}\, x \le \mathsf{succ}\, y$.*

Proof. This is a direct consequence of (10) and the correctness of Code. ◀

▶ **Lemma 32.** *Let $f : \mathbb{N} \xrightarrow{\le} \mathsf{Brw}$ be an increasing sequence and $x : \mathsf{Brw}$ a Brouwer tree such that $x < \mathsf{limit}\, f$. Then, there exists an $n : \mathbb{N}$ such that $x < f\, n$.*

Proof. By definition, $x < \mathsf{limit}\, f$ means $\mathsf{succ}\, x \le \mathsf{limit}\, f$. Using toCode together with the case (11), there exists an n such that $\mathsf{Code}\, (\mathsf{succ}\, x)\, (f\, n)$. Using fromCode, we get the result. ◀

▶ **Lemma 33.** *If f, g are increasing sequences such that $\mathsf{limit}\, f \le \mathsf{limit}\, g$, then f is simulated by g.*

Proof. For a given k, (13) tells us that there exists an n such that, after using fromCode, we have $f\, k \le g\, n$. ◀

▶ **Lemma 34.** *If f is an increasing sequence and x a Brouwer tree such that $\mathsf{limit}\, f \le \mathsf{succ}\, x$, then $\mathsf{limit}\, f \le x$.*

Proof. From (12), we have that $f\, k \le \mathsf{succ}\, x$ for every k. But since f is increasing, $\mathsf{succ}\, (f\, k) \le f\, (k + 1) \le \mathsf{succ}\, x$ for every k, hence by Lemma 31 $f\, k \le x$ for every k, and the result follows using the constructor $\le$-limiting. ◀

▶ **Theorem 1** (for Brouwer trees)**.** *Brw is a set, and $<$ as well as $\le$ are valued in propositions. Both $<$ and $\le$ are transitive, $<$ is irreflexive, and $\le$ is reflexive and antisymmetric.*

Proof. Brw and $\le$ are constructed to be a set and valued to be in propositions, respectively. $<$ is defined as an instance of $\le$ and thus valued in propositions. Transitivity for both relations is given by $\le$-trans.

Reflexivity of $\le$ holds by easy induction. Irreflexivity follows directly from wellfoundedness, which for $<$ is given in Theorem 4 below.

Antisymmetry of $\leq$ is shown with the characterisation of $\leq$ given by Lemma 30; we sketch how: Let x, y with $x \leq y$ and $y \leq x$ be given. We do nested induction, i.e. induction on both x and y. Since we are proving a proposition, we can disregard the cases for path constructors, giving us 9 cases in total, many of which are duplicates. We discuss the two most interesting cases:

- $x \equiv \mathsf{limit}\, f$ and $y \equiv \mathsf{succ}\, y'$: In that case, Lemma 34 and the assumed inequalities show $\mathsf{succ}\, y' \leq \mathsf{limit}\, f \leq y'$ and thus $y' < y'$, contradicting the wellfoundedness of $<$ (Theorem 4).
- $x \equiv \mathsf{limit}\, f$ and $y \equiv \mathsf{limit}\, g$: By Lemma 33, f and g simulate each other. By the constructor bisim, $x = y$. ◀

▶ **Theorem 2** (for Brouwer trees). *The relations $<$ and $\leq$ on* Brw *are extensional.*

Proof. Extensionality is trivial for a reflexive and antisymmetric relation, which covers $\leq$. For $<$, let x and y be two elements of Brw with the same set of smaller elements. As in the above proof, we can disregard the path constructors and get 9 cases. If x and y are built of different constructors, it is easy to derive a contradiction. For example, in the case $x \equiv \mathsf{limit}\, f$ and $y \equiv \mathsf{succ}\, y'$, we have $y' < y$ and thus $y' < \mathsf{limit}\, f$. By Lemma 32, there exists an n such that $y' < f\, n$, which in turn implies $y < f\,(n+1)$ and thus $y < \mathsf{limit}\, f$. By the assumed set of smaller elements, that means we have $y < y$, contradicting irreflexivity of $<$.

The other interesting case, $x \equiv \mathsf{limit}\, f$ and $y \equiv \mathsf{limit}\, g$, is easy. For all $k : \mathbb{N}$, we have $f\, k < f\,(k+1) \leq \mathsf{limit}\, f$ and thus $f\, k < \mathsf{limit}\, g$; by the constructor $\leq$-limiting, this implies $\mathsf{limit}\, f \leq \mathsf{limit}\, g$. By the symmetric argument and by antisymmetry of $\leq$, it follows that $\mathsf{limit}\, f = \mathsf{limit}\, g$. ◀

▶ **Theorem 4** (for Brouwer trees). *The relation $<$ on* Brw *is wellfounded.*

Proof. We need to prove that every y : Brw is accessible. When doing induction on y, the cases of path constructors are automatic as we are proving a proposition. From the remaining constructors, we only show the hardest case, which is that $y \equiv \mathsf{limit}\, f$. We have to prove that any given $x < \mathsf{limit}\, f$ is accessible. By Lemma 32, there exists an n such that $x < f\, n$, and the latter is accessible by the induction hypothesis. ◀

▶ **Theorem 5** (for Brouwer trees). Brw *and its relations satisfy:*

(A1) *$<$ is transitive and irreflexive;*
(A2) *$\leq$ is reflexive, transitive, and antisymmetric;*
(A3) *we have $(<) \subseteq (\leq)$ and $(< \circ \leq) \subseteq (<)$.*

Proof. The first two properties have been shown above. $(<) \subseteq (\leq)$ follows easily from Lemma 30 and $(< \circ \leq) \subseteq (<)$ by transitivity. ◀

▶ **Theorem 6** (for Brouwer trees). Brw *has a zero.*

Proof. The zero is given directly by the constructor zero, with $\leq$-zero witnessing its property. ◀

▶ **Theorem 8** (for Brouwer trees). Brw *has strong successors, and the successor function is both $<$- and $\leq$-monotone.*

Proof. The function is given by the constructor succ. It calculates successors by Lemma 7 and by definition. To verify that it is a strong successor we need to show that $x < \mathsf{succ}\, b$ implies $x \leq b$. But $x < \mathsf{succ}\, b$ is defined to mean $\mathsf{succ}\, x \leq \mathsf{succ}\, b$ which, by Lemma 31, is indeed equivalent to $x \leq b$. ◀

▶ **Theorem 9** (for Brouwer trees). Brw *has limits of increasing sequences.*

Proof. Directly given by the constructor limit in conjunction with the constructors $\leq$-cocone and $\leq$-limiting. ◀

▶ **Theorem 11** (for Brouwer trees). Brw *has classification.*

Proof. By Lemma 26 and Theorem 14 below. ◀

▶ **Theorem 14** (for Brouwer trees). Brw *satisfies classifiability induction.*

Proof. Since Brw has zero, successor, and limits of increasing sequences, all given by the respective constructor, the special case of induction for Brw where the goal is a proposition is exactly classifiability induction. ◀

▶ **Theorem 18** (for Brouwer trees). Brw *has addition, multiplication and exponentiation with every base, and they are all unique.*

Proof. The standard arithmetic operations on Brouwer trees can be implemented with the usual strategy well-known in the functional programming community, i.e. by recursion on the second argument. However, there are several additional difficulties which stem from the fact that our Brouwer trees enforce correctness.

Let us start with addition. The obvious definition is

$$\begin{aligned} x + \mathsf{zero} &:\equiv x \\ x + \mathsf{succ}\, y &:\equiv \mathsf{succ}\,(x + y) \\ x + \mathsf{limit}\, f &:\equiv \mathsf{limit}\,(\lambda k. x + f\, k). \end{aligned}$$

For this to work, we need to prove, mutually with the above definition, that the sequence $\lambda k.x + f\,k$ in the last line is still increasing, which follows from mutually proving that $+$ is monotone in the second argument, both with respect to $\leq$ and $<$. We also need to show that bisimilar sequences f and g lead to bisimilar sequences $x + f\,k$ and $x + g\,k$.

The same difficulties occur for multiplication $(\cdot)$, where they are more serious: If f is increasing (with respect to $<$), then $\lambda k.x + f\,k$ is not necessarily increasing, as x could be zero. What saves us is that it is *decidable* whether x is 0 (cf. Lemma 29); and if it is, the correct definition is $x \cdot \mathsf{limit}\, f :\equiv \mathsf{zero}$. If x is not 0, then it is at least 1 (another simple lemma for Brw), and the sequence *is* increasing. With the help of several lemmas that are all stated and proven mutually with the actual definition of $(\cdot)$, the mentioned decidability is the core ingredient which allows us to complete the construction. Exponentation x^y comes with similar caveats as multiplication, but works with the same strategy.

Our Agda formalisation contains proofs that addition, multiplication, and exponentation have the properties introduced in Section 4.4, and that they are all unique. ◀

The formalisation contains several other results which are omitted in the paper. Define $\iota : \mathbb{N} \to \mathsf{Brw}$ as the embedding of natural numbers into Brouwer trees, i.e. $\iota(0) :\equiv \mathsf{zero}$ and $\iota(n+1) :\equiv \mathsf{succ}\,(\iota(n))$, and let $\omega :\equiv \mathsf{limit}\,\iota$. We then have:

▶ **Lemma 35.** *Brouwer trees of the form ω^x are additive principal:*
(a) *if $a < \omega^x$ then $a + \omega^x = \omega^x$.*
(b) *if $a < \omega^x$ and $b < \omega^x$ then $a + b < \omega^x$.*

Proof. For (a), it is enough to show $a + \omega^x \leq \omega^x$ by monotonicity of $+$. We prove this by induction on a, making crucial use of Lemma 32. Statement (b) is an easy corollary of a) and strict monotonicity of $+$ in the second argument. ◀

▶ **Lemma 36.** *If $x > \mathsf{zero}$ and $n : \mathbb{N}$, then $\iota(n) \cdot \omega^x = \omega^x$.*

Proof. By induction on x. The base case $x = \mathsf{zero}$ is impossible by assumption. If $x = \mathsf{succ}\, y$, we can decide if $y = \mathsf{zero}$ or not. If it is, i.e. if $x = \mathsf{succ}\,\mathsf{zero}$, we prove $\iota(n) \cdot \omega = \omega$ by induction on n, and otherwise the goal follows straightforwardly from the induction hypothesis. In the limit case $x = \mathsf{limit}\, f$, we use that $\mathsf{limit}\, f = \mathsf{limit}\,(\lambda k.f\,(k+1))$, and that $f\,(k+1) > \mathsf{zero}$ for every increasing sequence f, in order for the induction hypothesis to apply. ◀

D Proofs about Extensional Wellfounded Orders

▶ **Theorem 1** (for extensional wellfounded orders)**.** Ord *is a set and the relations $<$ and $\leq$ are valued in propositions. In addition, $<$ and $\leq$ are transitive, $<$ is irreflexive, and $\leq$ is reflexive and antisymmetric. $<$ is trichotomous iff $\leq$ is connex iff* LEM *holds.*

Proof. All statements follow either from [40, Thm 10.3.20] or are included in Theorem 38 below. ◀

▶ **Theorem 2** (for extensional wellfounded orders)**.** *The relations $<$ and $\leq$ on* Ord *are extensional.*

Proof. Extensionality is trivial for a reflexive and antisymmetric relation, which covers $\leq$. Extensionality of $<$ is shown in [40, Thm 10.3.20]. ◀

▶ **Theorem 4** (for extensional wellfounded orders)**.** *The relation $<$ on* Ord *is wellfounded.*

Proof. See [40, Thm 10.3.20]. ◀

▶ **Theorem 5** (for extensional wellfounded orders)**.** Ord *and its relations satisfy:*
(A1) *$<$ is transitive and irreflexive;*
(A2) *$\leq$ is reflexive, transitive, and antisymmetric;*
(A3) *we have $(<) \subseteq (\leq)$ and $(< \circ \leq) \subseteq (<)$.*

Proof. The first two properties follow from Ord being an ordinal [40, Thm 10.3.20]. The first part of 3 is projection, the second holds by composition of functions. ◀

The property symmetric to $(< \circ \leq) \subseteq (<)$ fails for Ord:

▶ **Lemma 37.** *Let A, B, C :* Ord*. The statement that $A \leq B$ and $B < C$ implies $A < C$ holds if and only if excluded middle* LEM *does.*

Proof. If $A \simeq B_{/b}$ and $g : B \leq C$ then $A \simeq C_{/g(b)}$. Assuming LEM and $f : A \leq B$, $g : B < C$, there is a minimal $c : C$ not in the image of $g \circ f$ by [40, Theorem 10.4.3], and $A \simeq C_{/c}$. Conversely, let P be a proposition; it is an ordinal with the empty order. We have $\mathbf{1} \leq \mathbf{1} \uplus P$ and $\mathbf{1} \uplus P < \mathbf{1} \uplus P \uplus \mathbf{1}$, so by assumption $\mathbf{1} < \mathbf{1} \uplus P \uplus \mathbf{1}$. Now observe which component of the sum the bound is from: this shows if P holds or not. ◀

▶ **Theorem 6** (for extensional wellfounded orders)**.** Ord *has a zero.*

Proof. Zero is given as the empty type $\mathbf{0}$. ◀

▶ **Theorem 8** (for extensional wellfounded orders)**.** Ord *has strong successors, with the successor of A given as $A \uplus \mathbf{1}$, but monotonicity with respect to either $<$ or $\leq$ is equivalent to* LEM.

Proof. The definition of a bounded simulation implies $(X < Y) \leftrightarrow (X \uplus \mathbf{1} \leq Y)$, making Lemma 7 applicable. For the rest of the statement, see Theorem 38 below. ◀

▶ **Theorem 9** (for extensional wellfounded orders)**.** *If $F : \mathbb{N} \to \mathsf{Ord}$ is an increasing sequence of ordinals, then its limit $\lim F$ is the quotient $(\Sigma n : \mathbb{N}.Fn)/\sim$, where $(n, y) \sim (n', y')$ if and only if $(Fn)_{/y} \simeq (Fn')_{/y'}$, with $[(n, y)] \prec [(n', y')]$ if $(Fn)_{/y} < (Fn')_{/y'}$. If F is only weakly increasing (i.e. increasing with respect to $\leq$), the same formula defines the supremum of F.*

Proof. The construction of limits/suprema is from [40, Lem 10.3.22], where it is shown that the canonical function $l_n : Fn \to \lim F$ is a simulation. A family $g_n : Fn \leq X$ readily gives a function $(\Sigma n : \mathbb{N}.Fn) \to X$, which we extend to the quotient. Given $(n, y) \sim (n', y')$, let WLOG $n \leq n'$; hence there is $f : Fn \leq Fn'$. The condition $(Fn)_{/y} \simeq (Fn')_{/y'}$ implies $f\,y = y'$. As simulations are unique, we have $l_n = l_{n'} \circ f$, which means that (n, y) and (n', y') are indeed mapped to equal elements in $\lim F$, giving us a map $g : \lim F \to X$. As every g_n is a simulation, so is g. ◀

Note that the above proof explicitly uses that F is (at least weakly) increasing; we do not think that the more general construction in [40, Lem 10.3.22] can be shown constructively to give suprema in our sense.

▶ **Theorem 11** (for extensional wellfounded orders)**.** *If* Ord *has classification, the law of excluded middle is derivable.*

Proof. See Theorem 38. ◀

▶ **Theorem 14** (for extensional wellfounded orders)**.** *If* Ord *satisfies classifiability induction, we can derive the law of excluded middle.*

Proof. See Theorem 38. ◀

▶ **Theorem 18** (for extensional wellfounded orders)**.** Ord *has addition given by $A + B = A \uplus B$, and multiplication given by $A \cdot B = A \times B$, with the order reverse lexicographic, i.e. $(x, y) \prec (x', y')$ is defined to be $y \prec_B y' \uplus (y = y' \times x \prec_A x')$.*

Proof. The key observation is that a sequence of simulations $F0 \leq F1 \leq F2 \leq \ldots$ is preserved by adding or multiplying a constant *on the left*, i.e. we have $C \cdot F0 \leq C \cdot F1 \leq C \cdot F2$ (but note that adding a constant on the right fails, see Theorem 38 below). This allows us to use the explicit representation of suprema from Theorem 9. ◀

Many constructions that we have performed for Cnf and Brw are not possible for Ord, at least not constructively:

▶ **Theorem 38.** *Each of the following statements on its own implies the law of excluded middle (*LEM*):*

(a) *The successor $(\_ \uplus \mathbf{1})$ is $\leq$-monotone.*
(b) *The successor $(\_ \uplus \mathbf{1})$ is $<$-monotone.*
(c) *$<$ is trichotomous, i.e. $(X < Y) \uplus (X = Y) \uplus (X > Y)$.*
(d) *$\leq$ is connex, i.e. $(X \leq Y) \uplus (X \geq Y)$.*
(e) Ord *satisfies classifiability induction.*
(f) Ord *has classification.*

Assuming LEM*, the first four statements hold.*

Proof. We first show the chain LEM $\Rightarrow$ (a) $\Rightarrow$ (b) $\Rightarrow$ LEM.

LEM $\Rightarrow$ (a): Let $f : A \leq B$. Using LEM, there is a minimal $b : B \uplus \mathbf{1}$ which is not in the image of f [40, Theorem 10.4.3]. The simulation $A \uplus \mathbf{1} \leq B \uplus \mathbf{1}$ is given by $f \uplus b$.

(a) $\Rightarrow$ (b): Assume we have $A < B$. As in Theorem 8, this is equivalent to $A \uplus \mathbf{1} \leq B$. Assuming suc is $\leq$-monotone, we get $A \uplus \mathbf{2} \leq B \uplus \mathbf{1}$, and applying Lemma 7 once more, this is equivalent to $A \uplus \mathbf{1} < B \uplus \mathbf{1}$.

(b) $\Rightarrow$ LEM: Assume P is a proposition. We have $\mathbf{0} < \mathbf{1} \uplus P$. If suc is $<$-monotone, then we get $\mathbf{0} \uplus \mathbf{1} < \mathbf{1} \uplus P \uplus \mathbf{1}$. Observing if the simulation f sends $\mathsf{inr}(\star)$ to the P summand or not, we decide $P \uplus \neg P$.

Next, we show the chain LEM $\Rightarrow$ (c) $\Rightarrow$ (d) $\Rightarrow$ LEM, where the first implication is given by [40, Thm 10.4.1].

(c) $\Rightarrow$ (d): Each of the three cases of (c) gives us either $X \leq Y$ or $X \geq Y$ or both.

(d) $\Rightarrow$ LEM: Given $P : \mathsf{hProp}$, we compare $P \uplus P$ with $\mathbf{1}$. If $P \uplus P \leq \mathbf{1}$ then $\neg P$, while $\mathbf{1} \leq P \uplus P$ implies P.

Finally, we show (e) $\Rightarrow$ (f) $\Rightarrow$ LEM. The first implication is clear. For the second implication, observe that a classifiable proposition is either zero ($\mathbf{0}$) or a successor ($X \uplus \mathbf{1}$), as a limit is never a proposition. In both cases, we are done. ◀

E Proofs for Section 5

▶ **Theorem 19.** *The function* CtoB *preserves and reflects* $<$ *and* $\leq$*, i.e.,* $a < b \leftrightarrow \mathsf{CtoB}(a) < \mathsf{CtoB}(b)$*, and* $a \leq b \leftrightarrow \mathsf{CtoB}(a) \leq \mathsf{CtoB}(b)$*.*

Proof. We show the proof for $<$; each direction of the statement for $\leq$ is a simple consequence.

($\Rightarrow$) By induction on $a < b$. The case when $\boldsymbol{\omega}^a \boldsymbol{+} b < \boldsymbol{\omega}^c \boldsymbol{+} d$ because $a < c$ uses Lemma 35.

($\Leftarrow$) Assume $\mathsf{CtoB}(a) < \mathsf{CtoB}(b)$. If $a \geq b$, then $\mathsf{CtoB}(a) \geq \mathsf{CtoB}(b)$ by ($\Rightarrow$), conflicting the assumption. Hence $a < b$ by the trichotomy of $<$ on Cnf. ◀

▶ **Corollary 20.** *The function* CtoB *is injective.*

Proof. $\mathsf{CtoB}(a) = \mathsf{CtoB}(b)$ implies $\mathsf{CtoB}(a) \leq \mathsf{CtoB}(b)$ and thus, by Theorem 19, $a \leq b$. Analogously, one has $b \leq a$. Antisymmetry gives $a = b$. ◀

▶ **Theorem 21.** CtoB *commutes with addition, multiplication, and exponentiation with base* ω*.*

Proof. As an example, we show that CtoB commutes with addition, i.e., $\mathsf{CtoB}(a + b) = \mathsf{CtoB}(a) + \mathsf{CtoB}(b)$ for all $a, b : \mathsf{Cnf}$. The proof is carried out by induction on a, b. It is trivial when either of them is 0. Assume $a = \boldsymbol{\omega}^x \boldsymbol{+} u$ and $b = \boldsymbol{\omega}^y \boldsymbol{+} v$. If $x < y$, then $a + b = b$. We have also $\boldsymbol{\omega}^x < \boldsymbol{\omega}^y$, which implies $\omega^{\mathsf{CtoB}(x)} < \omega^{\mathsf{CtoB}(y)}$ by Theorem 19. Then by Lemma 35.a we have $\omega^{\mathsf{CtoB}(x)} + \omega^{\mathsf{CtoB}(y)} = \omega^{\mathsf{CtoB}(y)}$. By the same argument, from the fact $u < \boldsymbol{\omega}^y$ we derive $\mathsf{CtoB}(u) + \omega^{\mathsf{CtoB}(y)} = \omega^{\mathsf{CtoB}(y)}$. Therefore, both $\mathsf{CtoB}(a+b)$ and $\mathsf{CtoB}(a) + \mathsf{CtoB}(b)$ are equal to $\mathsf{CtoB}(b)$. If $y \leq x$, then $a + b = \boldsymbol{\omega}^x \boldsymbol{+} u + b$ by definition. By induction hypothesis, we have $\mathsf{CtoB}(u + b) = \mathsf{CtoB}(u) + \mathsf{CtoB}(b)$. Therefore, both $\mathsf{CtoB}(a + b)$ and $\mathsf{CtoB}(a) + \mathsf{CtoB}(b)$ are equal to $\omega^{\mathsf{CtoB}(x)} + \mathsf{CtoB}(u) + \mathsf{CtoB}(b)$.

The fact that CtoB commutes with multiplication can be proved with a similar argument of induction. Moreover, the proof relies on Lemma 36. Preservation of exponentiation with base ω holds by definition. ◀

▶ **Theorem 22.** *For all* $a : \mathsf{Cnf}$*, we have* $\mathsf{CtoB}(a) < \mathsf{limit}\,(\lambda k.\omega \uparrow\uparrow k)$*, where* $\omega \uparrow\uparrow 0 :\equiv \omega$ *and* $\omega \uparrow\uparrow (k+1) :\equiv \omega^{\omega \uparrow\uparrow k}$*.*

Proof. By induction on a. Using that $\varepsilon_0 = \omega^{\varepsilon_0} = \omega^{\omega^{\varepsilon_0}}$, in the step case we have $\omega^{\mathsf{CtoB}(a)} + \mathsf{CtoB}(b) < \varepsilon_0$ by Lemma 35, strict monotonicity of ω^{-}, and the induction hypothesis. ◀

▶ **Lemma 39.** *For $X : \mathsf{Ord}$ with $x : X$, the first projection $\mathsf{fst} : X_{/x} \to X$ is a simulation. If $x, y : X$ and $f : X_{/x} \to X_{/y}$ is a function, then f is a simulation if and only if $\mathsf{fst} \circ f = \mathsf{fst}$.*

Proof. Both properties required in the definition of a simulation are obvious in the case of fst. In the second sentence, if f is a simulation, then the equality follows from the uniqueness of simulations [40, Thm 10.3.16]. If the equality holds then, again, the two properties in the definition of a simulation are clear for f. ◀

▶ **Lemma 23.** *The function $\mathsf{BtoO} : \mathsf{Brw} \to \mathsf{Ord}$ is injective, and preserves $<$ and $\leq$.*

Proof. The first part (injectivity of BtoO) is a special case of the following statement: Given $X : \mathsf{Ord}$, the map $X \to \mathsf{Ord}$, $x \mapsto X_{/x}$ is injective. This is remarked just before [40, before 10.3.19]. We give a detailed proof:

Note that an equality $Y = Z$ in Ord gives rise to a canonical simulation $X \leq Y$ by path induction. Now, assume $x, y : X$ with $X_{/x} = X_{/y}$. We get $f : X_{/x} \leq X_{/y}$. By Lemma 39, f maps (z, p) to (z, q), with $q : z < y$; that is, every element below x is also below y. The symmetric statement follows by the symmetric argument, and injectivity of $x \mapsto X_{/x}$ by extensionality.

If $q : x \leq y$ in Brw, then the map $X_{/x} \to X_{/y}$, $(z, p) \to (z, p \cdot q)$ is a simulation by Lemma 39, thus BtoO preserves $\leq$. This implies that $<$ is preserved as well since $X < Y \leftrightarrow (X \uplus 1) \leq Y$. ◀

▶ **Theorem 24.** *Under the assumption of the law of excluded middle, the function $\mathsf{BtoO} : \mathsf{Brw} \to \mathsf{Ord}$ is a simulation.*

Proof. Given $b < \mathsf{BtoO}(a)$, we need to find a Brouwer tree $a' < a$ such that $\mathsf{BtoO}(a') = b$. Using LEM, we can choose a' to be the minimal Brouwer tree x such that $b \leq \mathsf{BtoO}(x)$. ◀

▶ **Theorem 25.** *If the map $\mathsf{BtoO} : \mathsf{Brw} \to \mathsf{Ord}$ is a simulation, then WLPO holds.*

Proof. Let a sequence s be given. We define the type family $S : \mathbb{N} \to \mathsf{hProp}$ by $S n :\equiv (s\, n = \mathsf{tt})$ and regard it as a family of orders. We have $\bigsqcup S < \mathbf{2}$. Observing that $\mathbf{2} \equiv \mathsf{BtoO}(\mathsf{succ}(\mathsf{succ}\,\mathsf{zero}))$ and using the assumption that BtoO is a simulation, we get $b < \mathsf{succ}(\mathsf{succ}\,\mathsf{zero})$ such that $\mathsf{BtoO}\, b = \bigsqcup S$. It is decidable whether b is zero or $\mathsf{succ}\,\mathsf{zero}$, and this determines if s is constantly ff or not. ◀