TYPE-THEORETIC APPROACHES TO ORDINALS

NICOLAI KRAUS, FREDRIK NORDVALL FORSBERG, AND CHUANGJIE XU

Abstract. In a constructive setting, no concrete formulation of ordinal numbers can simultaneously have all the properties one might be interested in; for example, being able to calculate limits of sequences is constructively incompatible with deciding extensional equality. Using homotopy type theory as the foundational setting, we develop an abstract framework for ordinal theory and establish a collection of desirable properties and constructions. We then study and compare three concrete implementations of ordinals in homotopy type theory: first, a notation system based on Cantor normal forms (binary trees); second, a refined version of Brouwer trees (infinitely-branching trees); and third, extensional well-founded orders.

Each of our three formulations has the central properties expected of ordinals, such as being equipped with an extensional and well-founded ordering as well as allowing basic arithmetic operations, but they differ with respect to what they make possible in addition. For example, for finite collections of ordinals, Cantor normal forms have decidable properties, but suprema of infinite collections cannot be computed. In contrast, extensional well-founded orders work well with infinite collections, but the price to pay is that almost all properties are undecidable. Brouwer trees, implemented as a quotient inductive-inductive type to ensure well-foundedness and extensionality, take the sweet spot in the middle by combining a restricted form of decidability with the ability to work with infinite increasing sequences.

Our three approaches are connected by canonical order-preserving functions from the "more decidable" to the "less decidable" notions, i.e. from Cantor normal forms to Brouwer trees, and from there to extensional well-founded orders. We have formalised the results on Cantor normal forms and Brouwer trees in cubical Agda, while extensional well-founded orders have been studied and formalised thoroughly by Escardó and his collaborators. Finally, we compare the computational efficiency of our implementations with the results reported by Berger.

Keywords: ordinal numbers, constructive mathematics, homotopy type theory, Cantor normal forms, Brouwer trees, extensional well-founded orders

Contents

1. Introduction

Ordinal numbers are an important tool in modern mathematics and proof theory, employed for example for showing termination of processes [27, 35], the semantics of inductive definitions [1, 30], and justifying recursion, as used in many papers by Berger and others on realisability and program extraction in the presence of induction and coinduction [7, 8, 9]. In these applications of ordinals, the metatheory is typically based on classical logic. The applications, however, are of interest also in constructive mathematics, and so, it would be of great benefit to develop constructive approaches to ordinals which are strong enough to handle such applications. In this article, we use type theory to develop such constructive approaches, and show that they also cover other important aspects of ordinals, such as their arithmetic theory, generalising the one of the natural numbers, and the existence of suprema of potentially unbounded sequences of ordinals.

1.1. **Three Approaches to Ordinal Numbers.** In classical set theory, there are various equivalent representations of ordinals, and, whenever one representation is more convenient than another, one can freely switch between them. However, switching to a different representation often requires the law of excluded middle or other constructively unavailable principles. Therefore, it is unsurprising that the situation is more challenging in a constructive setting. Different representations of ordinals are no longer equivalent, and it is easy to see that there cannot be a single formulation of ordinals that makes it possible to prove all the properties and perform all the constructions that the various applications require. For example, consider a binary sequence s, i.e. a function from the natural numbers into the set $\{0, 1\}$; if we had a formulation of ordinals that allowed us to calculate the limit x of the sequence s *and* decide extensional equality of x with 0, then this would amount to checking whether the sequence s is constantly 0. This, however, is exactly Bishop's *weak limited principle of omniscience* [11], an axiom that is generally not assumed in constructive mathematics.

When using or developing ordinals in a constructive setting, one is for this reason forced to make compromises and give up some desirable properties, and the choice will naturally depend on the anticipated applications. This explains why several different constructions of ordinals have been studied in the literature. It is also not unusual to see tailor-made inductive definitions replace applications of ordinals and transfinite induction, with the consequence that basic results are established over and over again, for each inductive definition. Instead, in this article, we will consider the following approaches to ordinals:

- One approach to develop ordinal theory is to use "syntactic" ordinal notation systems [17, 56, 60]. Such systems are popular with proof theorists, as their concrete character typically means that equality and the order relation on ordinals are decidable. However, truly infinitary operations such as calculating limits of infinite families of notations are usually not constructible.
- Another approach to ordinals, popular in the functional programming community and based on notation systems by Church [20] and Kleene [45], is to consider *Brouwer trees* $\mathcal{O}$ inductively generated by zero, successor, and a supremum constructor

$$\mathsf{sup} : (\mathbb{N} \to \mathcal{O}) \to \mathcal{O} \tag{1}$$

which forms a new tree for every countable sequence of trees [15, 21, 40]. 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. Of course, when allowing infinite sequences, extensional equality cannot be checked algorithmically.

- Yet another approach to ordinals is to consider collections of extensional well-founded orders satisfying transitivity, representing a variation on the classical set-theoretical axioms that is more suitable for a constructive treatment [61]. When pursuing a development of ordinals based on such orders without further conditions, one naturally gives up all non-trivial notions of decidability – it even becomes impossible to check whether a given order is zero, a successor, or a limit. Nevertheless, many operations can still be defined on the collection of all such orders, and properties such as well-foundedness can still be proven. This is also the notion of ordinal most closely related to the traditional notion, and thus, in a classical setting, the formulation which most obviously corresponds to the established literature.

Is it possible to specify what a proper definition of ordinals in a constructive setting is, how the different approaches fulfil the specification, which properties they lack, and how they are connected to each other? Can constructions and ideas be transported from one setting to another — e.g., do the arithmetic operations constructed for one notion of ordinals obey the same rules as the arithmetic operations defined for another notion? In order to develop one possible precise answer to these questions, we work in *homotopy type theory* [64] and suggest an abstract framework of ordinals in which the various desirable properties and operations can be formulated. We also study three concrete constructions, representing the three approaches above, which become instances of our general abstract framework.

The representative of the class of "syntactic" ordinal notation systems that we develop is based on Cantor normal forms using unlabelled binary trees. Our concrete definition ensures that we have no "junk" terms, i.e. each element of our type Cnf of Cantor normal forms denotes an actual ordinal. There are several reasonable ways in which such a type can be defined that can be shown to be equivalent to each other [53]; the construction that we choose defines Cnf as a subset of the type of binary trees.

When defining Brouwer trees as an inductive type with constructors for zero, successor, and a third constructor sup as in (1), it is a priori wishful thinking to call the last constructor a "supremum"; $\mathsf{sup}(s)$ 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 that would be supposed to represent the same supremum. Fortunately, this shortcoming can be fixed in homotopy type theory via a *higher* or *quotient inductive-inductive type* Brw, combining induction-induction with the idea of higher inductive types [22, 49]. While we naturally cannot derive decidable equality for the type Brw, we retain the possibility of classifying an ordinal as a zero, a successor or a limit.

Finally, the idea of extensional well-founded orders was transferred to the setting of homotopy type theory in the "HoTT book" [64, Chp 10], and significantly extended by Escardó and his collaborators [33]. The approach is to define Ord to be the type of pairs $(X, <)$, where the latter is a propositionally-valued, transitive, extensional, and well-founded relation. While Ord lacks all forms of non-trivial decidability, it is better suited for constructions involving infinite families of ordinals than Cnf or Brw.

All in all, each of the approaches above 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 well-founded orders the set-theoretic properties of ordinals. It turns out that there are canonical embeddings of the "more" into the "less decidable" notions, i.e. we have

TYPE-THEORETIC APPROACHES TO ORDINALS

Property	Cnf	Brw	Ord	More details
$x < y \leq z \to x < z$	✓	✓	✓	beginning of Section 4
$x \leq y < z \to x < z$	✓	✓	✗	
order is well-founded	✓	✓	✓	Section 4.1
order is extensional	✓	✓	✓	
has zero and successors	✓	✓	✓	Section 4.2
has finite suprema (binary joins)	✓	✗	✓	
has limits of strictly increasing $\mathbb{N}$-sequences	✗	✓	✓	
has suprema of small families	✗	✗	✓	
can classify as zero, successor, or limit	✓	✓	✗	
has addition	✓	✓	✓	Section 4.3
has multiplication	✓	✓	✓	
has (partial) exponentiation	✓	✓		
has subtraction	✓	✗*	✗	
has division	✓	✗	✗	
$x < \omega$ is decidable	✓	✓	✗	Section 4.4
$x < y$ is decidable	✓	✗*	✗	
$x \leq y$ splits as $(x < y) \uplus (x = y)$	✓	✗*	✗	
is trichotomous	✓	✗*	✗	
computational efficiency	(good)	(medium)	(none)	Section 9

* Each of these properties is equivalent to Bishop's *limited principle of omniscience* (LPO), cf. Section 2.2.

TABLE 1. Summary of how the three notions of ordinals compare.

functions from Cantor normal forms to Brouwer trees and from Brouwer trees to extensional well-founded orders. We study whether, or under which conditions, these functions preserve arithmetic operations, commute with limits, and are simulations. Inspired by Berger's results on the computational efficiency of implementations of ordinals [6], we also show how Cnf and Brw perform when implemented in Cubical Agda [66]. Note that we cannot compute in a meaningful way with Ord. A summary of how the three notions of ordinals compare can be found in Table 1.

1.2. **Related Work.** The development of ordinals in constructive mathematics has a rich history [15, 20, 37, 41, 45, 51, 52]. As mentioned above, one well-known constructive notion of ordinal is given by extensional well-founded relations. However, the transitivity

$$x \leq y < z \to x < z \qquad (2)$$

fails because it implies excluded middle. Taylor [61, 62] recovers it by introducing plumpness, which essentially restricts to the subclass for which the property (2) holds hereditarily. We do not explicitly study plump ordinals in this paper, but the transitivity (2) holds for both Cantor normal forms and Brouwer trees. In their recent work [23], Coquand, Lombardi and Neuwirth develop another constructive theory of ordinals. They start with a structure of certain linear orders, called $\mathfrak{F}$-orders, to describe the desirable properties of ordinals including the transitivity (2). Then they inductively construct a set **Ord** of ordinals and prove that **Ord** is initial in the category of $\mathfrak{F}$-orders. In this way, they show that their constructive ordinals satisfy the desirable properties constructively. Our abstract axiomatic framework introduced in Section 4 is similar to their structure of $\mathfrak{F}$-orders. But it is more general (e.g., no assumption like (2)) because it is for relating and comparing our three different approaches to ordinals.

Several formalisations of ordinals and ordinal notation systems exist in the literature. Escardó and his collaborators develop many results of extensional well-founded relations in homotopy type theory, and have formalised them in Agda [33]. The theory of ordinals below ε_0 based on various representations has been developed in some formal systems. For the representation of Cantor normal forms with coefficients, Manolios and Vroon [50] work in ACL2, Castéran and Contejean [19] and Grimm [39] in Coq, and Shinkarov [58] in Agda. Blanchette, Fleury and Traytel [12] work with the representation of hereditary multisets in Isabelle/HOL. One of our approaches to ordinals is based on Cantor normal forms without coefficients. In this paper, we additionally prove that the arithmetic operations on Cantor normal forms are uniquely correct with respect to our abstract axiomatisation.

In the work on ordinals below ε_0 [19, 39, 53, 58], one derives/implements transfinite induction directly from the selected representation of ordinals. Berger [6] instead extracts a program from Gentzen's proof [38] of transfinite induction up to ε_0. Gentzen's proof involves nesting of implications of bounded depth. Therefore, the extracted program contains functionals of arbitrarily high types (with respect to the finite type structure). Using the extracted program, one obtains higher type primitive recursive definitions of the fast growing hierarchy and tree ordinals of height below ε_0. Berger compares the transfinite recursive implementation of the Hardy hierarchy and the one via the extracted higher type program, and observes that the latter is much faster. Inspired by Berger's experiment, we compare the computational efficiency of our notions of ordinals in Section 9.

1.3. **Agda Formalisation.** As indicated above, we have formalised the material on Cantor normal forms and Brouwer trees in cubical Agda [66]:

- Git repository of the Agda code: `bitbucket.org/nicolaikraus/constructive-ordinals-in-hott/`
- DOI of the archived Agda code: 10.5281/zenodo.7657456
- HTML rendering: `cj-xu.github.io/agda/type-theoretic-approaches-to-ordinals/`

While the development in the Git repository listed above is ongoing and not restricted to the current paper, the HTML rendering and the archived version are snapshots of the repository at the time of writing. To complement the formalisation, we also refer to Escardó's Agda library [33] that consists of many results on extensional well-founded orders in HoTT. In the paper, we have marked theorems whose proofs we have formalised, or partly formalised, using the symbols ⚙⚙ and ⚙ respectively; they are also clickable links to the corresponding machine-checked statement in the HTML rendering of our Agda code.

Our formalisation builds on the agda/cubical library [63] and type checks using Agda version 2.6.3. It uses the `{-# TERMINATING #-}` pragma to work around a limitation of the termination checker of Agda: recursive calls hidden under a propositional truncation are not seen to be structurally smaller. While we believe that such recursive calls when proving a proposition are justified by Dijkstra's eliminator presentation [29], it would be non-trivial to reduce our mutual definitions to eliminators.

1.4. **History of this Paper.** An earlier, short version of this paper appeared in the proceedings of the conference *Mathematical Foundations of Computer Science* [47]. New in this version is a thorough discussion of the constructive aspects and decidability results (Section 4.4) of the various approaches to ordinals via *constructive taboos* (cf. Section 2.2). In particular, we rigorously study the decidable and undecidable properties of Brouwer trees (Section 4.4) and connect the preservation of limits of the embedding $\mathsf{Cnf} \to \mathsf{Brw}$ with Markov's principle (cf. Theorem 67 and Lemma 68).

In addition to providing a complete Agda formalisation (except for extensional well-founded orders), we benchmark the efficiency of computing with Cantor normal forms and Brouwer trees (Section 9), inspired by Berger's benchmarking of ordinal recursive versus higher type programs [6].

2. Preliminaries

In this section, we introduce concepts and notation that we are going to use in the rest of the paper.

2.1. **Concepts of Homotopy Type Theory.** We work in and assume basic familiarity with homotopy type theory (HoTT), i.e. Martin-Löf type theory extended with higher inductive types and the univalence axiom [64]. 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 $x : A$. We further write $A \leftrightarrow B$ for "if and only if", i.e. for functions in both directions $A \to B$ and $B \to A$. 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*, $\mathsf{isProp}(A)$, 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*, $\mathsf{isSet}(A)$, if $\mathsf{isProp}(x = y)$ for every $x, y : A$. We write $\mathsf{hSet} = \Sigma(A : \mathcal{U}).\mathsf{isSet}(A)$ for the type of sets, again with the first projection implicit when necessary.

We denote *propositional truncation* of a type A by $\|A\|$, which is the smallest proposition with a function from A. In particular, by $\exists(x : A).B(x)$, we mean the propositional truncation of $\Sigma(x : A).B(x)$, i.e., we have $\exists(x : A).B(x) : \mathsf{hProp}$, 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 $*$, and $\mathbf{2}$ for the type with two elements ff and tt.

2.2. **Constructive Taboos.** When we are unable to perform a certain construction or prove a theorem formulated as a type A, we want to understand why it is seemingly impossible to define an element of A. An obvious approach is attempting to derive a contradiction from the assumption that A holds, i.e. prove $A \to \mathbf{0}$, a type that we also denote by $\neg A$ ("not A"). However, this will often not be possible either in a constructive setting, since many interesting statements A are consistent and may actually turn out to be true in a classical theory. Therefore, we may be forced to replace the empty type by a less ambitious goal B; something that is known from models to not be provable in the type theory we work in, or something that is simply generally undesirable in a constructive setting. We call such a type B a *constructive taboo*; it is also sometimes known as a Brouwerian counterexample [14].

Obviously, the empty type $\mathbf{0}$ is a taboo in all interesting (non-trivial) settings, so it is technically also a constructive taboo. Another very prominent constructive taboo is the *law of excluded middle* LEM, stating that every proposition is either true or false:

$$\mathsf{LEM} :\equiv \forall(P : \mathsf{hProp}).P \uplus \neg P. \tag{3}$$

Several other taboos that we consider talk about binary sequences. The first is Bishop's *limited principle of omniscience* LPO [11], stating that every binary sequence is either constantly ff or somewhere tt:

$$\mathsf{LPO} :\equiv \forall(s : \mathbb{N} \to \mathbf{2}).(\forall n.s_n = \mathsf{ff}) \uplus (\exists n.s_n = \mathsf{tt}). \tag{4}$$

The weakened version, known as the *weak limited principle of omniscience* WLPO, states that it is decidable whether a sequence is constantly ff:

$$\mathsf{WLPO} :\equiv \forall(s : \mathbb{N} \to \mathbf{2}).(\forall n.s_n = \mathsf{ff}) \uplus \neg(\forall n.s_n = \mathsf{ff}). \tag{5}$$

Finally, *Markov's principle* MP says that, if a sequence is not constantly ff, then it is tt somewhere:

$$\mathsf{MP} :\equiv \forall(s : \mathbb{N} \to \mathbf{2}).\neg(\forall n.s_n = \mathsf{ff}) \to (\exists n.s_n = \mathsf{tt}). \tag{6}$$

We always have $(\forall n.s_n = \mathsf{ff}) \leftrightarrow \neg(\exists n.s_n = \mathsf{tt})$. If we view a binary sequence $s : \mathbb{N} \to \mathbf{2}$ as representing a semidecidable property (cf. [5]), then LPO says that every semidecidable property is decidable ($P \uplus \neg P$), while WLPO says that every semidecidable property is weakly decidable ($\neg P \uplus \neg\neg P$), and MP postulates that every semidecidable property is stable ($\neg\neg P \to P$). It is thus not surprising, and well known, that we have:

Lemma 1 (). *The limited principle of omniscience is as strong as the weak limited principle of omniscience and Markov's principle combined:*

$$\mathsf{LPO} \leftrightarrow \mathsf{WLPO} \times \mathsf{MP}. \tag{7}$$

□

Note that the distinction between $\exists$ and Σ is inessential in the formulation of the above taboos. This is shown by the following lemma (see e.g. Escardó [32] and Escardó and Xu [34, §3.1]):

Lemma 2 (). *For any sequence $s : \mathbb{N} \to \mathbf{2}$, we have*

$$(\exists n.s_n = \mathsf{tt}) \to \Sigma(n : \mathbb{N}).s_n = \mathsf{tt}. \tag{8}$$

In particular, if we assume LPO*, then a given sequence is either constantly* ff *or we concretely get an $n : \mathbb{N}$ where it is not.*

Proof. Refining the type of n such that $s_n = \mathsf{tt}$ to the type of *minimal* n with this property, we get a proposition and can eliminate out of the truncation. In detail, we construct a function

$$\forall n.s_n = \mathsf{tt} \to (\Sigma(n : \mathbb{N}).(s_n = \mathsf{tt}) \times \Pi(k < n).s_k = \mathsf{ff}) \tag{9}$$

that searches the minimal index where a sequence is positive. Using that the target is a proposition, we precompose this function with the eliminator of the truncation. Finally, we compose with the projection function forgetting that n is minimal. □

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 [64], 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

$$\alpha = \omega^{\beta_1} + \omega^{\beta_2} + \cdots + \omega^{\beta_n} \text{ with } \beta_1 \geq \beta_2 \geq \cdots \geq \beta_n \tag{10}$$

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 [17, 19, 39, 53]. Let $\mathcal{T}$ be the type of unlabeled binary trees, i.e. the inductive type with suggestively named constructors $0 : \mathcal{T}$ and $\omega^{-} \mathbin{\boldsymbol{+}} - : \mathcal{T} \times \mathcal{T} \to \mathcal{T}$. Let the relation $<$ be the *hereditary lexicographical order*, i.e. generated by the following clauses:

$$0 < \omega^{a} \mathbin{\boldsymbol{+}} b \tag{11}$$

$$a < c \to \omega^{a} \mathbin{\boldsymbol{+}} b < \omega^{c} \mathbin{\boldsymbol{+}} d \tag{12}$$

$$b < d \to \omega^{a} \mathbin{\boldsymbol{+}} b < \omega^{a} \mathbin{\boldsymbol{+}} d. \tag{13}$$

We have the map $\mathsf{left} : \mathcal{T} \to \mathcal{T}$ defined by $\mathsf{left}(0) :\equiv 0$ and $\mathsf{left}(\omega^{a} \mathbin{\boldsymbol{+}} b) :\equiv a$ which gives us the left subtree (if it exists) of a tree. A tree is a *Cantor normal form* (CNF) if, for every $\omega^{s} \mathbin{\boldsymbol{+}} t$ 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 (10). For instance, both trees $1 :\equiv \omega^{0} \mathbin{\boldsymbol{+}} 0$ and $\omega :\equiv \omega^{1} \mathbin{\boldsymbol{+}} 0$ are CNFs. Formally, the predicate isCNF is defined inductively by the two clauses

$$\mathsf{isCNF}(0) \tag{14}$$

$$\mathsf{isCNF}(s) \to \mathsf{isCNF}(t) \to \mathsf{left}(t) \leq s \to \mathsf{isCNF}(\omega^{s} \mathbin{\boldsymbol{+}} t). \tag{15}$$

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 trees* (or *Brouwer ordinal trees*) in functional programming are 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

$$\mathsf{zero} : \mathsf{Brw} \tag{16}$$

$$\mathsf{succ} : \mathsf{Brw} \to \mathsf{Brw} \tag{17}$$

$$\mathsf{limit} : (\mathbb{N} \xrightarrow{\leq} \mathsf{Brw}) \to \mathsf{Brw} \tag{18}$$

$$\mathsf{bisim} : (f\, g : \mathbb{N} \xrightarrow{\leq} \mathsf{Brw}) \to f \approx^{\leq} g \to \mathsf{limit}\, f = \mathsf{limit}\, g, \tag{19}$$

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 mutually defined relation $\leq$ is inductively defined by the following constructors, 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:

$$\leq\text{-zero} \quad : \quad \mathsf{zero} \leq x \tag{20}$$
$$\leq\text{-trans} \quad : \quad x \leq y \to y \leq z \to x \leq z \tag{21}$$
$$\leq\text{-succ-mono} : \quad x \leq y \to \mathsf{succ}\, x \leq \mathsf{succ}\, y \tag{22}$$
$$\leq\text{-cocone} \quad : \quad (k : \mathbb{N}) \to x \leq f(k) \to x \leq \mathsf{limit}\, f \tag{23}$$
$$\leq\text{-limiting} \quad : \quad (\forall k. f(k) \leq x) \to \mathsf{limit}\, f \leq x \tag{24}$$

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 $\leq$-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 strictly increasing sequences, we can avoid the representation of zero or successor ordinals as limits (as otherwise e.g. $a = \mathsf{limit}\,(\lambda i.a)$). If one wishes to drop this restriction, it will be necessary to strengthen the bisim constructor to witness antisymmetry — however, we found that version of Brw significantly harder to work with; see Section 6.7 for a short discussion. Another question is whether adding the constructor $\leq$-trans explicitly is necessary since, even without including $\leq$-trans in the construction, it might be possible to derive transitivity of $\leq$ anyway. We do not know the answer to that question.

3.3. **Transitive, Extensional and Well-Founded Orders.** The third notion of ordinals that we consider is the one studied in the HoTT book [64]. This is the notion which is closest to the classical definition of an ordinal as a set with a well-founded, trichotomous, 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*, stating that any two elements of X with the same predecessors are equal.

Concretely, an ordinal in the sense of the HoTT book [64, Def 10.3.17] is a type[1] X together with a relation $\prec : X \to X \to \mathsf{hProp}$ which is *transitive*, *extensional*, and *well-founded* (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 the HoTT book [64, 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 $X_{/x} :\equiv \Sigma(y : X).y \prec x$. Again following the HoTT book [64, Def 10.3.19], a simulation $f : X \leq Y$ is *bounded* if we have $y : Y$ such that f induces an equivalence

[1] Note that the HoTT book [64, Def 10.3.17] asks for X to be a set, but Escardó [33] proved that this follows from the rest of the definition, and we therefore drop this requirement.

$X \simeq Y_{/y}$. 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? Compared to the previous section, we go up one level of abstraction and consider an arbitrary set A together with relations $<$ and $\leq$ valued in propositions:

$$A \quad : \mathsf{hSet} \tag{25}$$

$$(\_<\_) : A \to A \to \mathsf{hProp} \tag{26}$$

$$(\_\leq\_) : A \to A \to \mathsf{hProp}. \tag{27}$$

The types Cnf, Brw, Ord with their relations are concrete implementations of such a triple $(A, <, \leq)$.[2] In the current section, we discuss the various constructions that a concrete implementation may or may not allow, and list the main results (see the "summary boxes" below and on the next pages); in Sections 5 to 7, we discuss Cnf, Brw, and Ord respectively in detail, and prove the precise theorems.

For Cnf, the relation $\leq$ is the reflexive closure of $<$, but the analogous statement is not constructively provable for Brw and Ord. In the current section, we make the following basic assumptions:

(A1) $<$ is transitive ($x < y \to y < z \to x < z$) and irreflexive ($\neg(x < x)$);
(A2) $\leq$ is reflexive ($x \leq x$), transitive, and antisymmetric ($x \leq y \to y \leq x \to x = y$);
(A3) we have $(<) \subseteq (\leq)$ and $(< \circ \leq) \subseteq (<)$.

On top of these assumptions, we can now consider additional properties that we would expect ordinals to have. The third condition (A3) means that $(x < y) \to (x \leq y)$ and $(x < y) \to (y \leq z) \to (x < z)$. We do not assume the "symmetric" variation

$$(\leq \circ <) \subseteq (<), \tag{28}$$

which is true for Cnf and Brw, but only holds for Ord iff LEM holds. This constructive failure is known and can be seen as motivation for *plump* ordinals [59, 61].

Proving that $\leq$ for Brw is antisymmetric is challenging because of the path constructors in the inductive-inductive definition of Brouwer trees. Of course, this difficulty arises as a consequence of our chosen definition for Brw, and other definitions would make antisymmetry easy; but unsurprisingly, such alternative definitions simply shift the difficulties to other places, see Section 6.7.

In Sections 5 to 7, we will prove in detail which of the properties discussed here hold for Cnf, Brw, and Ord. In the current section, we very briefly summarise these results in boxes such as the one below. While some of the stated properties are original results of the current paper, others are known and stated for comparison only; the references included in the boxes lead to the precise theorems and proofs or citations.

Summary of results

The assumptions (A1), (A2), and (A3) are satisfied for Cnf, Brw, and Ord. The property $(\leq \circ <) \subseteq (<)$ holds for Cnf and Brw, but is equivalent to LEM for Ord.

Precise statements: Thm 18 (Cnf); Cor 37 (Brw); Lem 57 and Cor 59 (Ord).

For the rest of Section 4, we assume that $(A, <, \leq)$ is given and satisfies the conditions above.

[2] Note that we do not require A to live in a specified universe, as Ord is larger than Cnf or Brw.

4.1. **Extensionality and Well-Foundedness.** Following the HoTT book [64, Def 10.3.9], we call a relation $\prec$ *extensional* if, for all $a, b : A$, we have

$$(\forall c. c \prec a \leftrightarrow c \prec b) \to a = b. \tag{29}$$

Assumption (A2) implies that $\leq$ is extensional: given a and b such that $c \leq a \leftrightarrow c \leq b$ for every c, we have $a \leq a$ by reflexivity, and hence $a \leq b$ by assumption. Similarly, we get $b \leq a$, and hence $a = b$ by antisymmetry. In contrast, extensionality is not guaranteed for $<$, but we will show that it holds for our three instances Cnf, Brw, and Ord. This is non-trivial in the case of Brw — note that it fails for the "naive" version of Brw, where the path constructor bisim is missing.

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

$$\mathit{access} : (a : A) \to ((b : A) \to b < a \to \mathsf{Acc}(b)) \to \mathsf{Acc}(a). \tag{30}$$

An element $a : A$ is called *accessible* if $\mathsf{Acc}(a)$, and $<$ is *well-founded* if all elements of A are accessible. It is well known that the following induction principle can be derived from the inductive presentation [64]:

Lemma 3 (, transfinite induction)**.** *Let $<$ be well-founded 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 all our use cases in this paper, P will be a mere property (i.e. a family of propositions), although the induction principle is valid even without this assumption.

As a standard sample application, we show that the classical formulation of well-foundedness is a consequence:

Lemma 4 ()**.** *If $<$ is well-founded, then there is no infinite decreasing sequence:*

$$\neg \left(\Sigma(f : \mathbb{N} \to A).(i : \mathbb{N}) \to f(i+1) < f(i) \right). \tag{31}$$

In particular, there is no cycle $a_0 < a_1 < \ldots < a_n < a_0$. For $n \equiv 0$, this says that $<$ is irreflexive.

Proof. We apply Lemma 3 with the property P given by

$$P(a) \;:\equiv\; \neg\Sigma(f : \mathbb{N} \to A).(f\,0 = a) \times ((i : \mathbb{N}) \to f(i+1) < f(i)). \tag{32}$$

To show the induction step, assume a sequence f with $f(0) = a$ is given. Then, the sequence $\lambda i. f(i+1)$ gives a contradiction by the induction hypothesis. □

From the global assumptions that A is a set and $<$ is irreflexive as well as valued in propositions, we get that $x < y$ and $x = y$ are mutually exclusive propositions. Therefore, we get the following observation for the reflexive closure:

Lemma 5 ()**.** *The reflexive closure of $<$, given by $(x < y) \uplus (x = y)$, is valued in propositions.* □

Summary of results

For each of Cnf, Brw, and Ord, the relation $<$ is extensional and well-founded.

Precise statements: Thms 18 and 19 (Cnf); Thms 35 and 38 (Brw); Thm 58 (Ord).

Note that the results stated so far in particular mean that Cnf and Brw can be seen as elements of Ord themselves.

4.2. **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 family of ordinals.

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

$$\mathsf{is\text{-}zero}(a) :\equiv \forall b. a \leq b, \tag{33}$$

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

We say that a is a *successor* of b if it is the least element strictly greater[3] than b:

$$(a \ \mathsf{is\text{-}suc\text{-}of} \ b) :\equiv (a > b) \times \forall x > b. a \leq x. \tag{34}$$

We say that $(A, <, \leq)$ *has successors* if there is a function $s : A \to A$ which calculates successors, i.e. such that $\forall b. s(b)$ $\mathsf{is\text{-}suc\text{-}of}$ 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 (A3).

Lemma 6. *A function $s : A \to A$ 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:

$$(a \ \mathsf{is\text{-}str\text{-}suc\text{-}of} \ b) :\equiv a \ \mathsf{is\text{-}suc\text{-}of} \ b \times \forall x < a. x \leq b. \tag{35}$$

We say that A *has strong successors* if there is $s : A \to A$ which calculates strong successors, i.e. such that $\forall b. s(b)$ $\mathsf{is\text{-}str\text{-}suc\text{-}of}$ b. The additional information contained in a strong successor plays an important role in our technical development.

Finally, we consider the *suprema* or, synonymously, *least upper bounds* of families of ordinals. Given a type X and $f : X \to A$, an element $a : A$ is the supremum of f if it is at least as large as every $f(x)$, and if any b with this property is at least as large as a:

$$(a \ \mathsf{is\text{-}sup\text{-}of} \ f) :\equiv (\forall x. f(x) \leq a) \times (\forall b. (\forall x. f(x) \leq b) \to a \leq b). \tag{36}$$

We say that $(A, <, \leq)$ *has suprema of X-indexed families* if there is a function $\sqcup : (X \to A) \to A$ which calculates suprema, i.e. such that $(f : X \to A) \to (\sqcup f)$ $\mathsf{is\text{-}sup\text{-}of}$ f. Note that the supremum is unique if it exists, i.e. the type of suprema is propositional for a given pair (X, f). Both the properties "calculating suprema" and "having suprema" are propositions. If $(A, <, \leq)$ has suprema of **2**-indexed families, we also say that A has *binary joins*. Unless explicitly specified, in the following we will consider suprema of $\mathbb{N}$-indexed families only.

Instead of considering functions f without further structure, we can ask for f to be a morphism of (partial) orders. Of particular interest is the case of $\leq$*-monotone* $\mathbb{N}$-indexed sequences, i.e. functions $f : \mathbb{N} \to A$ such that $f_n \leq f_{n+1}$. We also consider strictly increasing $\mathbb{N}$-indexed sequences satisfying the condition $f_n < f_{n+1}$. Note that 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 for strictly increasing $\mathbb{N}$-sequences, and in this case, we call a the *limit* of f:

$$\_ \ \mathsf{is\text{-}}\mathbb{N}\mathsf{\text{-}lim\text{-}of} \ \_ : A \to (\mathbb{N} \xrightarrow{<} A) \to \mathcal{U} \tag{37}$$

$$a \ \mathsf{is\text{-}}\mathbb{N}\mathsf{\text{-}lim\text{-}of} \ (f, q) :\equiv a \ \mathsf{is\text{-}sup\text{-}of} \ f. \tag{38}$$

[3]Note that $>$ is the obvious symmetric notation for $<$; it is *not* a newly assumed relation.

We say that A *has limits* if there is a function $\mathsf{limit} : (\mathbb{N} \xrightarrow{<} A) \to A$ that calculates limits of strictly increasing $\mathbb{N}$-sequences. Note that Cnf cannot have limits since one can construct a sequence (see Theorem 70) which comes arbitrarily close to ε_0. The question becomes more interesting if we consider *bounded* sequences, i.e. sequences f with $b : \mathsf{Cnf}$ such that $f_i < b$ for all $i : \mathbb{N}$.

Summary of results

All our notions of ordinals have zeroes and strong successors. Ord is the only considered notion that constructively has suprema of arbitrary small families — this was proven by de Jong and Escardó [25]. Cnf only has suprema of finite families, which Brw does not have constructively; on the other hand, Brw has suprema of strictly increasing $\mathbb{N}$-indexed sequences. Monotonicity of the successor function holds for Cnf and Brw, but not constructively for Ord.

Precise statements: Lem 22 and Thm 29 (Cnf); Lem 39, Cor 40, and Thm 55 (Brw); Lem 60 and Thm 63 (Ord).

4.2.2. *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 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 zero in (33). We now also define what it means for $a : A$ to be a strong successor or a limit of a strictly increasing $\mathbb{N}$-indexed sequence:

$$\mathsf{is\text{-}str\text{-}suc}(a) :\equiv \Sigma(b : A).(a\ \mathsf{is\text{-}str\text{-}suc\text{-}of}\ b) \tag{39}$$

$$\mathsf{is\text{-}}\mathbb{N}\mathsf{\text{-}lim}(a) :\equiv \exists f : \mathbb{N} \xrightarrow{<} A.\, a\ \mathsf{is\text{-}}\mathbb{N}\mathsf{\text{-}lim\text{-}of}\ f. \tag{40}$$

In addition, we say that a is a general limit if it is the supremum of the set of all elements below a (and there exists at least one such element). As in Section 3.3, we write $A_{/a} :\equiv \Sigma(b : A).b < A$ for this type. We have the first projection $\mathsf{fst} : A_{/a} \to A$ and define:

$$\mathsf{is\text{-}general\text{-}lim}(a) :\equiv \|A_{/a}\| \times (a\ \mathsf{is\text{-}sup\text{-}of}\ \mathsf{fst}). \tag{41}$$

Remark 7 (⚙). One can also consider to define a to be a general limit if a is the supremum of *some* small and inhabited family of elements below a, i.e., if there exists a small inhabited type X and $f : X \to A$ such that $f(x) < a$ for every $x : X$, and a $\mathsf{is\text{-}sup\text{-}of}$ f. Note that every general limit in this sense is also a general limit in the sense of (41), and if A is small, then the two notions are equivalent. However, the type of transitive, extensional, and well-founded orders in particular is not small, and so this definition is different from (41) for Ord, as $\mathsf{fst} : \mathsf{Ord}_{/X} \to \mathsf{Ord}$ is a *large* family. We do not know whether the relevant part of Theorem 63 holds for this notion of general limits, i.e., whether LEM implies that every X is either zero or a successor or the limit of a small family.

All of $\mathsf{is\text{-}zero}(a)$, $\mathsf{is\text{-}str\text{-}suc}(a)$, $\mathsf{is\text{-}}\mathbb{N}\mathsf{\text{-}lim}(a)$, and $\mathsf{is\text{-}general\text{-}lim}(a)$ are propositions. This is true even though $\mathsf{is\text{-}str\text{-}suc}(a)$ is defined without a propositional truncation because, if a is the strong successor of both b and b', we have $b \leq b'$ and $b' \leq b$, implying $b = b'$ by antisymmetry.

Lemma 8 (⚙). *Any $a : A$ can be at most one out of {zero, strong successor, limit}, and in a unique way. In other words, the type* $\mathsf{is\text{-}zero}(a) \uplus \mathsf{is\text{-}str\text{-}suc}(a) \uplus \mathsf{is\text{-}general\text{-}lim}(a)$ *is a proposition. Similarly, the type* $\mathsf{is\text{-}zero}(a) \uplus \mathsf{is\text{-}str\text{-}suc}(a) \uplus \mathsf{is\text{-}}\mathbb{N}\mathsf{\text{-}lim}(a)$ *is a proposition.*

Proof. As mentioned above, all considered summands are propositions. What we have to check is that they mutually exclude each other. Note that $\mathsf{is}\text{-}\mathbb{N}\text{-}\mathsf{lim}(a)$ implies $\mathsf{is}\text{-}\mathsf{general}\text{-}\mathsf{lim}(a)$, which means it suffices to consider the case of $\mathsf{is}\text{-}\mathsf{zero}(a) \uplus \mathsf{is}\text{-}\mathsf{str}\text{-}\mathsf{suc}(a) \uplus \mathsf{is}\text{-}\mathsf{general}\text{-}\mathsf{lim}(a)$.

Since the goal is a proposition, we can assume that we are given b and $b_0 < a$ 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 $\mathsf{fst} : A_{/a} \to A$, the same argument (with b replaced by b_0) applies. Finally, assume that a is the strong successor of b and the limit of fst. These assumptions show that b is an upper bound of fst, thus we get $a \leq b$. Together with $b < a$, this gives the contradiction $b < b$ as above. □

We say that an element of A is *weakly classifiable* if it is zero or a strong successor or a general limit, and *classifiable* if it is zero or a strong successor or a limit of a strictly increasing $\mathbb{N}$-indexed sequence. We say $(A, <, \leq)$ *has (weak) classification* if every element of A is (weakly) classifiable.[4] By Lemma 8, $(A, <, \leq)$ has classification exactly if the type $\mathsf{is}\text{-}\mathsf{zero}(a) \uplus \mathsf{is}\text{-}\mathsf{str}\text{-}\mathsf{suc}(a) \uplus \mathsf{is}\text{-}\mathbb{N}\text{-}\mathsf{lim}(a)$ is contractible, in the jargon of homotopy type theory.

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

Definition 9 (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

$$\mathsf{is}\text{-}\mathsf{zero}(a) \to P(a) \tag{42}$$

$$(a \ \mathsf{is}\text{-}\mathsf{str}\text{-}\mathsf{suc}\text{-}\mathsf{of}\ b) \to P(b) \to P(a) \tag{43}$$

$$(a \ \mathsf{is}\text{-}\mathbb{N}\text{-}\mathsf{lim}\text{-}\mathsf{of}\ f) \to (\forall i. P(f_i)) \to P(a), \tag{44}$$

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

Note that classifiability induction does *not* ask that there are functions that computes successors or limits. The following is immediate:

Corollary 10 (, of Lemma 8)**.** *If $(A, <, \leq)$ satisfies classifiability induction, then it has classification.* □

For the reverse direction, we need a further assumption:

Theorem 11 ()**.** *Assume $(A, <, \leq)$ has classification and satisfies the principle of transfinite induction. Then $(A, <, \leq)$ satisfies the principle of classifiability induction.*

Proof. With the assumptions of the statement and (42),(43), and (44), we need to show $\forall a. P(a)$. By transfinite induction, it suffices to show

$$(\forall b < a. P(b)) \to P(a) \tag{45}$$

for some fixed a. By classification, we can consider three cases. If $\mathsf{is}\text{-}\mathsf{zero}(a)$, then (42) gives us $P(a)$, which shows (45) 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 (45) quantifies over; therefore, this is implied by (43). Similarly, if $\mathsf{is}\text{-}\mathbb{N}\text{-}\mathsf{lim}(a)$, the assumption of (45) gives $\forall i. P(f_i)$, thus (44) gives $P(a)$. □

[4]As the terminology suggests, we focus on limits of increasing sequences and classification, while weak classification only plays a very minor role. The reason is that none of our notions of ordinals has weak classification without having classification, and the latter is easier to work with.

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 (A1), (A2), (A3) that are satisfied for Brw and Ord.

Summary of results

Cnf and Brw have classification and satisfy classifiability induction, while Ord does not, constructively. Ord has weak classification if an only if LEM holds.

Precise statements: Thm 27 (Cnf); Thm 41 (Brw); Thm 63 (Ord).

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

Definition 12 (having addition)**.** We say that $(A, <, \leq)$ *has addition* if there is a function $+ : A \to A \to A$ which satisfies the following properties:

$$\mathsf{is\text{-}zero}(a) \to c + a = c \tag{46}$$

$$a \ \mathsf{is\text{-}suc\text{-}of}\ b \to d \ \mathsf{is\text{-}suc\text{-}of}\ (c + b) \to c + a = d \tag{47}$$

$$a \ \mathsf{is\text{-}\mathbb{N}\text{-}lim\text{-}of}\ f \to b \ \mathsf{is\text{-}sup\text{-}of}\ (\lambda i.c + f_i) \to c + a = b \tag{48}$$

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

Note that (48) 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 (48) has to talk about the supremum, not the limit.

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

Definition 13 (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:

$$\mathsf{is\text{-}zero}(a) \to c \cdot a = a \tag{49}$$

$$a \ \mathsf{is\text{-}suc\text{-}of}\ b \to c \cdot a = c \cdot b + c \tag{50}$$

$$a \ \mathsf{is\text{-}\mathbb{N}\text{-}lim\text{-}of}\ f \to b \ \mathsf{is\text{-}sup\text{-}of}\ (\lambda i.c \cdot f_i) \to c \cdot a = b \tag{51}$$

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

Definition 14 (having exponentiation)**.** Assume A has addition $+$ and multiplication $\cdot$. We say that A *has exponentiation with base* c if we have a function $c^{-} : A \to A$

that satisfies the following properties:

$$\text{is-zero}(b) \to a \text{ is-suc-of } b \to c^b = a \tag{52}$$

$$a \text{ is-suc-of } b \to c^a = c^b \cdot c \tag{53}$$

$$a \text{ is-}\mathbb{N}\text{-lim-of } f \to \neg\text{is-zero}(c) \to b \text{ is-sup-of } c^{f_i} \to c^a = b \tag{54}$$

$$a \text{ is-}\mathbb{N}\text{-lim-of } f \to \text{is-zero}(c) \to c^a = c \tag{55}$$

A has unique exponentiation with base c if it has unique addition and multiplication, and if c^- is unique.

Let us now define subtraction, an operation that can be specified using addition. Note that there is more than one canonical choice: Given numbers a and b with $a \leq b$, we can either require their difference c to satisfy $a + c = b$ or $c + a = b$ (or even both), but only the first option (*left subtraction*) is in line with the specification for addition.[5]

Definition 15 (having subtraction)**.** We say that $(A, <, \leq)$ *has subtraction* if it has addition $+$, and a function $- : (b : A) \to (a : A) \to (p : a \leq b) \to A$, written $b -_p a$, such that $a + (b -_p a) = b$. We say that *A has unique subtraction* if it has unique addition and there is exactly one function $-$ with these properties.

Completely analogously, it would be possible to specify division and logarithm. Such constructions are further discussed in Section 5 below in the context of Cantor normal forms, but play otherwise no role in this paper.

Summary of results

Cnf uniquely has addition, multiplication, exponentiation with base ω, subtraction, and division. Brw uniquely has addition, multiplication, and exponentiation. Further, Brw has subtraction (necessarily unique) if and only if LPO holds. Ord has addition and multiplication, but subtraction is available if and only if LEM holds.

Precise statements: Thms 25 and 28 and Lem 24 (Cnf); Thms 42 and 45 (Brw); Thms 61 and 62 (Ord).

4.4. **Constructive Aspects.** The main differences between our three versions of constructive ordinals are their varying degrees of decidability of certain properties. As described in the introduction, we view Cantor normal forms as a notion where almost everything is decidable, while basically nothing is decidable for extensional well-founded orders, and Brouwer trees sit in the sweet spot in the middle. In this section, we want to make this intuition precise.

We say that a proposition[6] P is *decidable* if we have either P or $\neg P$, i.e.

$$\text{Dec}(P) :\equiv P \uplus \neg P, \tag{56}$$

and *stable* if its double negation is as strong as P,

$$\text{Stable}(P) :\equiv \neg\neg P \to P. \tag{57}$$

Of course, decidable propositions are always stable.

Given a set A, we can then ask whether equality is stable $(\forall(x, y : A).\text{Stable}(x = y))$ or even decidable $(\forall(x, y : A).\text{Dec}(x = y))$. Slightly weakening the properties we can, for a given $x_0 : A$, ask whether equality is locally stable $(\forall(y : A).\text{Stable}(x_0 = y))$

[5] If a is a limit, then $c + a$ cannot be a successor, and vice versa; in other words, requiring $c + a = b$ would imply that the difference between a limit and a successor cannot exist.

[6] While these definitions do not require P to be a proposition but work for any type, we only use them for propositions.

or decidable $(\forall(y : A).\mathsf{Dec}(x_0 = y))$. If the set comes with relations $<$, $\leq$, we can ask the same questions for these. Moreover, we can ask whether $\leq$ *splits*,

$$\mathsf{Splits}(A, <, \leq) :\equiv \forall(x, y : A).(x \leq y) \to (x < y) \uplus (x = y). \tag{58}$$

A relation $\leq$ is *connex* if $(a \leq b) \uplus (b \leq a)$, and a relation $<$ is *trichotomous* if $(a < b) \uplus (a = b) \uplus (b < a)$. Note that these two properties are very strong; under mild additional assumptions, they imply most or all of the other discussed properties. As an example, we have:

Lemma 16 (). *If $(A, <, \leq)$ satisfies the assumptions (A1) and $<$ is trichotomous, then assumption (A3) holds if and only if $\leq$ splits.*

Proof. We only show the direction "only if". Assume $x \leq y$. By trichotomy, we have $x < y$ or $x = y$ or $y < x$. In the first or second case, we are done. In the last case, (A3) implies $y < y$, contradicting (A1). □

When we work with concrete implementations of types of ordinals, it would of course be great to have a formulation that combines as many desirable properties as possible. However, we cannot have certain properties at the same time, as demonstrated by the following no-go theorem:

Theorem 17 (). *Assume that $(A, <, \leq)$ has zero, successors, and limits of strictly increasing sequences. If A has decidable equality, then* WLPO *holds.*

Proof. Zero z and a successor function s allow us to define a canonical strictly increasing function $\iota : \mathbb{N} \to A$. Let us write ω for the limit of this sequence.

Let $t : \mathbb{N} \to \mathbf{2}$ be a binary sequence. We construct a sequence

$$t^{\uparrow} : \mathbb{N} \to A \tag{59}$$

by

$$t^{\uparrow}\, 0 :\equiv z \tag{60}$$

$$t^{\uparrow}\, (n+1) :\equiv \begin{cases} \omega & \textit{if } n \textit{ is minimal such that } t_n = \mathsf{tt} \\ s(t^{\uparrow}\, n) & \textit{else.} \end{cases} \tag{61}$$

We call $t^{\uparrow}$ the *jumping sequence* of t as it "jumps" as soon as a tt is discovered in the sequence t. It is easy to see that $t^{\uparrow}$ is strictly increasing. By assumption, it thus has a limit $j : A$.

We claim that $j = \omega$ if and only if $\forall i.t_i = \mathsf{ff}$. If $j = \omega$, then $t_i = \mathsf{tt}$ leads to a contradiction for any i, thus we have $\forall i.t_i = \mathsf{ff}$. Vice versa, if $\forall i.t_i = \mathsf{ff}$, then $j = \omega$ by construction.

Therefore, if the equality $j = \omega$ is decidable, then so is the property $\forall i.t_i = \mathsf{ff}$. □

Theorem 17 means that a "perfectly convenient" implementation of ordinals cannot exist in a constructive world without WLPO, and any implementation with zero and successors will have to sacrifice either decidable equality or limits. In this paper, the three implementations demonstrating these choices are Cnf, Brw, and Ord.

Summary of results

Everything is decidable for Cnf (as long as no infinite families of ordinals are involved), $\leq$ splits and is connex, and $<$ is trichotomous. The situation is very different for Ord, where most properties that can be formulated using the above concepts imply or are equivalent to LEM.

Brw is the most interesting case: Many of the discussed properties are equivalent to LPO, including decidability of the relations, splitting of $\leq$, and trichotomy. At the same time, it is decidable whether x : Brw is finite, and equalities/inequalities are decidable on the subtype of finite Brouwer tree ordinals. Local equality at ω is decidable if and only if WLPO holds, but local equality at $\omega \cdot 2$ is already equivalent to LPO. While local equality at ω is stable, local equality at $\omega \cdot 2$ implies MP.

Precise statements: Thm 18 (Cnf); Thms 46, 48 to 51 and 53 (Brw); Thms 63 and 64 (Ord).

5. Cantor Normal Forms

Ordinal notation systems based on Cantor normal forms (with or without coefficients) have been widely studied [6, 19, 26, 28, 39, 53, 58, 60]. In this section, we recall the well-known results of Cantor normal forms, adapted for our chosen representation Cnf defined in Section 3.1. We additionally prove that the arithmetic operations on Cnf are uniquely correct with respect to our axiomatisation (Theorems 25 and 28), which has not been verified for Cantor normal forms previously, as far as we know.

As mentioned above, Cnf provides a decidable formulation of ordinals in the following sense.

Theorem 18 (⚙). *Cnf is a set with decidable equality. The relations $<$ and $\leq$ are valued in propositions, decidable, extensional, and transitive. In addition, $<$ is irreflexive and trichotomous, while $\leq$ is reflexive, antisymmetric, connex, and splits. If $x \leq y$ and $y < z$, then $x < z$ follows.*

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 the 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. □

Using Lemma 16, the above theorem shows that $(\mathsf{Cnf}, <, \leq)$ is an instantiation of the triple $(A, <, \leq)$ considered in Section 4.

We recall the following well-foundedness result of CNFs, which can be found in Nordvall Forsberg and Xu [53, Thm 5.1] and in Grimm [39, §2.3].

Theorem 19 (⚙). *The relation $<$ is well-founded.* □

By Lemma 3, we obtain transfinite induction for CNFs. We next move on to arithmetic on CNF, which is defined using decidability of $<$.

Definition 20 (Addition and multiplication of CNFs)**.** We define addition and multiplication as follows.[7]

$$0 + b :\equiv b \tag{62}$$

$$a + 0 :\equiv a \tag{63}$$

$$(\omega^a \boldsymbol{+} c) + (\omega^b \boldsymbol{+} d) :\equiv \begin{cases} \omega^b \boldsymbol{+} d & \text{if } a < b \\ \omega^a \boldsymbol{+} (c + \omega^b \boldsymbol{+} d) & \text{otherwise} \end{cases} \tag{64}$$

$$0 \cdot b :\equiv 0 \tag{65}$$

$$a \cdot 0 :\equiv 0 \tag{66}$$

$$a \cdot (\omega^0 \boldsymbol{+} d) :\equiv a + a \cdot d \tag{67}$$

$$(\omega^a \boldsymbol{+} c) \cdot (\omega^b \boldsymbol{+} d) :\equiv (\omega^{a+b} \boldsymbol{+} 0) + (\omega^a \boldsymbol{+} c) \cdot d \quad \text{if } b \neq 0 \tag{68}$$

The above operations are well-defined and have the expected ordinal arithmetic properties:

Lemma 21 (⚙)**.** *If a, b are CNFs, then so are $a + b$ and $a \cdot b$. Both operations are associative and strictly increasing in the right argument. Moreover, $(\cdot)$ is distributive on the left, i.e., $a \cdot (b + c) = a \cdot b + a \cdot c$.* □

Recall that we write 1 as abbreviation for $\omega^0 \boldsymbol{+} 0$. It is immediate to check that:

Lemma 22 (⚙)**.** *0 is a zero (in the sense of (33)), and $\lambda a.a + 1$ gives strong successors that are $<$- and $\leq$-monotone.* □

We also have:

Definition 23 (Exponentiation with base ω)**.** We define the CNF ω by $\omega :\equiv \omega^1 \boldsymbol{+} 0$ and the exponentiation ω^a of CNF a with base ω by $\omega^a :\equiv \omega^a \boldsymbol{+} 0$.

It is easy to show that $\omega^{(-)}$ is exponentiation with base ω in the sense of Definition 14. To show that $(+)$ is addition and $(\cdot)$ is multiplication in the sense of Definitions 12 and 13, we need their inverse operations subtraction and division.

Lemma 24 (⚙)**.** *For all CNFs a, b,*

(i) if $a \leq b$, then there is a CNF c such that $a + c = b$ and thus we denote c by $b - a$;
(ii) if $b > 0$, then there are CNFs c and d such that $a = b \cdot c + d$ and $d < b$.

Proof. For (i), we define subtraction $(-)$ as follows:

$$0 - b :\equiv 0 \tag{69}$$

$$a - 0 :\equiv a \tag{70}$$

$$(\omega^a \boldsymbol{+} c) - (\omega^b \boldsymbol{+} d) :\equiv \begin{cases} 0 & \text{if } a < b \\ c - d & \text{if } a = b \\ \omega^a \boldsymbol{+} c & \text{if } a > b. \end{cases} \tag{71}$$

See our formalisation for the proof of correctness. The proof of (ii) 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:

[7]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.

- $a = \omega^u + u'$ 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 $a = \omega^u + u' = \omega^{v+(u-v)} + u' = b \cdot \omega^{(u-v)} + b \cdot c' + d = b \cdot c + d$.
- $a = \omega^u + u'$ 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 25 (⚙). *Cnf has addition $(+)$, multiplication $(\cdot)$ and exponentiation $\omega^{(-)}$ with base ω.*

Proof. We show the limit case for $(+)$, and refer to our formalisation for the rest. Suppose a is the limit of f. The goal is to show that $c + a$ is the supremum (and thus the limit) of $\lambda i.c + fi$. We know $c + fi \leq c + a$ for all i because $(+)$ is increasing in the right argument (Lemma 21). It remains to prove that if $c + fi \leq x$ for all i then $c + a \leq x$. Thanks to Lemma 24(i), we have $fi \leq x - c$ for all i and thus $a \leq x - c$ because a is the limit of f. Therefore, we have $c + a \leq c + (x - c) = x$. □

We conjecture that Cnf has exponentiation with *arbitrary* base.[8] Specifically, we have constructed an operation $(-)^{(-)}$ and attempted to show a *logarithm* lemma: for any CNFs $a > 0$ and $b > 1$, there are CNFs x, y and z such that $a = b^x \cdot y + z$ and $0 < y < b$ and $z < b^x$.

All the arithmetic operations of CNFs are unique. An easy way to prove this fact is to use classifiability induction (Definition 9) which we obtain as follows — note that we can classify a CNF as a limit, even if we cannot compute limits of CNFs.

Lemma 26 (⚙). *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 a strictly 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 $s_i :\equiv \omega^a + r_i$.

The sequence s is known as the *fundamental sequence* of the CNF x. □

The construction of fundamental sequences for limit CNFs is standard and well known. For example, Grimm has developed it in Coq [39, §2.5].

Theorem 27 (⚙). *Cnf has classification and satisfies classifiability induction.*

Proof. Since Cnf has decidable equality, being zero and being a successor are both decidable. Then Lemma 26 shows that Cnf has classification. We then get classifiability induction from Theorem 11. □

We use classifiability induction to prove the uniqueness of the arithmetic operations.

Theorem 28 (⚙). *The operations of addition, multiplication and exponentiation with base ω on Cnf are unique.*

[8]The formalised proof is work in progress at the time of submission of this paper.

Proof. We sketch the proof for the uniqueness of addition and refer to our formalisation for the rest. Assume that $(+')$ is also an addition operation on CNFs. The goal is to show that $x + y = x +' y$ for all x and y. We use classifiability induction on y. The zero- and successor-cases are trivial. When y is a limit, we use the fact that $(+)$ preserves suprema, i.e., if a is a supremum of a sequence f, then $c + a$ is a supremum of the sequence $\lambda i.c + f_i$. □

We can check if a CNF is a limit and construct the fundamental sequence for limit CNFs. However, we cannot compute suprema or limits in general.

Theorem 29 (⚙). Cnf *does not have suprema or limits. Assuming the law of exclude middle* (LEM)*,* Cnf *has suprema (and thus limits) of arbitrary* bounded *sequences. If* Cnf *has limits of bounded strictly increasing sequences, then the weak limited principle of omniscience* (WLPO) *is derivable.*

Proof. To show that Cnf does not have suprema or limits, we construct the following counterexample. Let $\omega \uparrow\uparrow$ be a sequence of CNFs defined by $\omega \uparrow\uparrow 0 :\equiv \omega$ and $\omega \uparrow\uparrow (k+1) :\equiv \omega^{\omega \uparrow\uparrow k}$. If it has a limit, say x, then any CNF a is strictly smaller than x, including x itself. But this is in contradiction with irreflexivity.

For the second part, we use Theorem 10.4.3 from the HoTT book [64, Thm 10.4.3] which we recall as Lemma 56 in Section 7. It states that, assuming LEM, $(A, \prec)$ is an extensional well-founded order if and only if every nonempty subset $B \subseteq A$ has a least element. Given a bounded sequence f with bound b, we consider the subset $P : \mathsf{Cnf} \to \mathsf{hProp}$ of all the CNFs that are upper bounds of f. This subset contains at least b and is thus nonempty. We already have that $<$ on Cnf is extensional and well-founded. Therefore, if we assume LEM, then P has a least element which is a supremum of f.

On the other hand, the proof of Theorem 17 demonstrates that, if sequences bounded by $\omega \cdot 2$ have limits, then WLPO holds. □

6. Brouwer Trees

We now consider the construction of Brouwer trees in Section 3.2 in more detail: the type Brw was defined mutually with the relation $\leq$, and we defined $x < y$ as $\mathsf{succ}\, x \leq y$. The elimination principles for such a *quotient inductive-inductive construction* [29] are on an intuitive level explained in the HoTT book (e.g. in Chapter 11.3 [64, Chp 11.3]), and a full specification as well as further explanations are given by Altenkirch, Capriotti, Dijkstra, Kraus and Nordvall Forsberg [2] and Kaposi and Kovács [42, 43, 44].

We want to elaborate on the arguments that are required to establish the results listed in Section 4. Many proofs are very easy, for example the property (A3) of "mixed transitivity" is (almost) directly given by the constructor $\leq$-trans (cf. our formalisation); the property (28) is true as well, with an only slightly less direct argument. When we prove a propositional property by induction on a Brouwer tree, we only need to consider cases for point constructors, and multiple properties already follow from this. Below, we focus on the more difficult arguments and explain some of the more involved proofs.

6.1. Distinction of Constructors.

To start with, we need to prove that the point constructors of Brw are distinguishable, e.g. that we have $\neg(\mathsf{zero} = \mathsf{succ}\, x)$ — 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 $\mathsf{isZero} : \mathsf{Brw} \to \mathsf{hProp}$ in the proof of Lemma 30 below) works.

Lemma 30 (⚙). *The constructors of* Brw *are distinguishable, i.e. one can construct proofs of* $\mathsf{zero} \neq \mathsf{succ}\, x$, $\mathsf{zero} \neq \mathsf{limit}\, f$, *and* $\mathsf{succ}\, x \neq \mathsf{limit}\, f$.

Proof. We show how to distinguish zero and $\mathsf{succ}\, x$; the other parts are shown in the same way. Setting

$$\mathsf{isZero}\ \mathsf{zero} :\equiv \mathbf{1} \tag{72}$$
$$\mathsf{isZero}\ (\mathsf{succ}\, x) :\equiv \mathbf{0} \tag{73}$$
$$\mathsf{isZero}\ (\mathsf{limit}\, f) :\equiv \mathbf{0} \tag{74}$$

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

6.2. **Codes Characterising $\leq$.** Antisymmetry of $\leq$ as well as well-foundedness and extensionality of $<$ are among the technically most difficult results about Brw that we present in this paper. They are also the properties that would most easily fail with the "wrong" definition of Brw. To see the difficulty, let us for example consider well-foundedness of $<$: Given a strictly increasing sequence f, we have to show that $\mathsf{limit}\, f$ is accessible, i.e. that any given $x < \mathsf{limit}\, f$ is accessible. However, the induction hypothesis only tells us that every $f\, k$ is accessible. Thus, we want to show that there exists a k such that $x < f\, k$, but doing this directly does not seem possible.

We use a strategy corresponding to the *encode-decode method* [48] and define a type family

$$\mathsf{Code} : \mathsf{Brw} \to \mathsf{Brw} \to \mathsf{hProp} \tag{75}$$

which has the *correctness* properties

$$\mathsf{toCode} : \quad x \leq y \to \mathsf{Code}\, x\, y \tag{76}$$
$$\mathsf{fromCode} : \mathsf{Code}\, x\, y \to x \leq y, \tag{77}$$

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

$$\mathsf{Code}\ \ \mathsf{zero}\ \ \_ :\equiv \mathbf{1} \tag{78}$$
$$\mathsf{Code}\ \ (\mathsf{succ}\, x)\ \ \mathsf{zero} :\equiv \mathbf{0} \tag{79}$$
$$\mathsf{Code}\ \ (\mathsf{succ}\, x)\ (\mathsf{succ}\, y) :\equiv \mathsf{Code}\ x\ y \tag{80}$$
$$\mathsf{Code}\ \ (\mathsf{succ}\, x)\ (\mathsf{limit}\, f) :\equiv \exists n.\mathsf{Code}\, (\mathsf{succ}\, x)\, (f\, n) \tag{81}$$
$$\mathsf{Code}\ \ (\mathsf{limit}\, f)\ \ \mathsf{zero} :\equiv \mathbf{0} \tag{82}$$
$$\mathsf{Code}\ \ (\mathsf{limit}\, f)\ (\mathsf{succ}\, y) :\equiv \forall k.\mathsf{Code}\, (f\, k)\, (\mathsf{succ}\, y) \tag{83}$$
$$\mathsf{Code}\ \ (\mathsf{limit}\, f)\ (\mathsf{limit}\, g) :\equiv \forall k.\exists n.\mathsf{Code}\, (f\, k)\, (g\, n) \tag{84}$$

The part of the definition of Code given above is easy enough; the tricky 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 not difficult; 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 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:

$$\mathsf{toCode}: \ x \leq y \to \mathsf{Code}\, x\, y \tag{85}$$

$$\mathsf{Code\text{-}trans} : \mathsf{Code}\, x\, y \to \mathsf{Code}\, y\, z \to \mathsf{Code}\, x\, z \tag{86}$$

$$\mathsf{Code\text{-}refl} : \mathsf{Code}\, x\, x \tag{87}$$

$$\mathsf{Code\text{-}cocone} : \mathsf{Code}\, x\, (f k) \to \mathsf{Code}\, x\, (\mathsf{limit}\, f) \tag{88}$$

$$\mathsf{Code\text{-}succ\text{-}incr\text{-}simple} : \mathsf{Code}\, x\, (\mathsf{succ}\, x) \tag{89}$$

After the mutual definition is complete, we can separately prove $\mathsf{fromCode} : \mathsf{Code}\, x\, y \to x \leq y$. The complete construction is presented in the Agda formalisation.

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

Lemma 31 (⚙). *For $x, y : \mathsf{Brw}$, we have $x \leq y \leftrightarrow \mathsf{succ}\, x \leq \mathsf{succ}\, y$.*

Proof. This is a direct consequence of (80) and the correctness of Code. ☐

Lemma 32 (⚙). *Let $f : \mathbb{N} \xrightarrow{\leq} \mathsf{Brw}$ be a strictly 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 $\mathsf{succ}\, x \leq \mathsf{limit}\, f$. Using toCode together with the case (81), 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 strictly increasing sequences with $\mathsf{limit}\, f \leq \mathsf{limit}\, g$, then f is simulated by g.*

Proof. For every $k : \mathbb{N}$, (84) tells us that there exists an $n : \mathbb{N}$ such that, after using fromCode, we have $f\, k \leq g\, n$. ☐

Lemma 34 (⚙). *If f is a strictly increasing sequence and x a Brouwer tree such that $\mathsf{limit}\, f \leq \mathsf{succ}\, x$, then $\mathsf{limit}\, f \leq x$. Dually, limits are closed under successors: if $x < \mathsf{limit}\, f$ then also $\mathsf{succ}\, x < \mathsf{limit}\, f$.*

Proof. From (83), we have that $f\, k \leq \mathsf{succ}\, x$ for every k. But since f is increasing, $\mathsf{succ}\, (f\, k) \leq f\, (k+1) \leq \mathsf{succ}\, x$ for every k, hence by Lemma 31 $f\, k \leq x$ for every k, and the result follows using the constructor $\leq$-limiting. For the second statement, if $x < \mathsf{limit}\, f$ then by Lemma 32 we have $x < f\, n$ for some n, and since f is increasing, $\mathsf{succ}\, x < f\, (n+1) < \mathsf{limit}\, f$. ☐

An alternative proof of Lemma 31, which does not rely on the machinery of codes, is given in our formalisation.

6.3. **Antisymmetry, Well-Foundedness, and Extensionality.** With the help of the consequences of the characterisation of $\leq$ shown above, we can show multiple non-trivial properties of Brw and its relations. Regarding well-foundedness, we can now complete the argument sketched above:

Theorem 35 (⚙). *The relation $<$ is well-founded.*

Proof. We need to prove that every $y : \mathsf{Brw}$ 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 when $y \equiv \mathsf{limit}\, f$. 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. ☐

Next we show that $\leq$ is antisymmetric, i.e. if $x \leq y$ and $y \leq x$ then $x = y$.

Theorem 36 (⚙). *The relation $\leq$ is antisymmetric.*

Proof. Let x, y with $x \leq y$ and $y \leq x$ be given. We do nested induction. As before, 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 well-foundedness of $<$.
- $x \equiv \mathsf{limit}\, f$ and $y \equiv \mathsf{limit}\, g$: By Lemma 33, f and g simulate each other. By the constructor bisim, $x = y$. □

Corollary 37. $(\mathsf{Brw}, <, \leq)$ *satisfies the assumptions (A1), (A2), and (A3), i.e. it is an instantiation of the abstract triple* $(A, <, \leq)$ *discussed in Section 4. Furthermore, the symmetric variation of the second half of assumption (A3) holds for* $(\mathsf{Brw}, <, \leq)$*, i.e., if* $x \leq y$ *and* $y < z$*, then* $x < z$*.*

Proof. The only non-immediate properties are antisymmetry of $\leq$ (Theorem 36) and irreflexivity of $<$ (Theorem 35 and Lemma 4). That $x \leq y < z$ implies $x < z$ follows directly from the definition of $a < b$ as $\mathsf{succ}\, a \leq b$, transitivity of $\leq$, and monotonicity of succ. □

Finally we can show that $<$ is extensional, i.e. that Brouwer trees with the same predecessors are equal.

Theorem 38. *The relation* $<$ *is extensional.*

Proof. Let x and y be two elements of Brw with the same set of smaller elements. As in the above proof, we can consider 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 well-foundedness.

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 $\leq$-$\mathsf{limiting}$, this implies $\mathsf{limit}\, f \leq \mathsf{limit}\, g$. By the symmetric argument and by antisymmetry of $\leq$, it follows that $\mathsf{limit}\, f = \mathsf{limit}\, g$. □

6.4. **Classifiability.** Classifiability is straightforward for Brw, as the point constructors of the data type exactly corresponds to zero, successors and limits.

Lemma 39. Brw *has zero, strong successors, and limits of strictly increasing sequences, and each part is given by the corresponding constructor.* □

Proof. Most of these claims are easy. To verify that succ 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 $x \leq b$. □

By the definition of $<$ for Brw, we have:

Corollary 40 (of Lemma 31)**.** *The strong successor of* Brw *is* $<$*- and* $\leq$*-monotone.*

Hence we can now observe that the special case of induction for Brw where the goal is a proposition is exactly classifiability induction, and by Corollary 10, Brw has classification. This proves the following theorem:

Theorem 41. Brw *has classification and satisfies classifiability induction.* □

6.5. **Arithmetic of Brouwer Trees.** 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

$$x + \mathsf{zero} \quad :\equiv \quad x \tag{90}$$
$$x + \mathsf{succ}\, y \quad :\equiv \quad \mathsf{succ}\,(x + y) \tag{91}$$
$$x + \mathsf{limit}\, f \quad :\equiv \quad \mathsf{limit}\,(\lambda k. x + f\,k). \tag{92}$$

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: Even if f is increasing (with respect to $<$), then $\lambda k.\, x \cdot f\,k$ is not necessarily increasing, as x could be zero. What saves us is that it is *decidable* whether x is zero (cf. Section 6.1); and if it is, the correct definition is $x \cdot \mathsf{limit}\, f :\equiv \mathsf{zero}$. If x is not zero, then it is at least $\mathsf{succ}\,\mathsf{zero}$ (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:

$$x \cdot \mathsf{zero} \quad :\equiv \quad \mathsf{zero} \tag{93}$$
$$x \cdot \mathsf{succ}\, y \quad :\equiv \quad (x \cdot y) + x \tag{94}$$
$$x \cdot \mathsf{limit}\, f \quad :\equiv \quad \begin{cases} \mathsf{zero} & \text{if } x = \mathsf{zero} \\ \mathsf{limit}\,(\lambda k. x \cdot f\,k). & \text{otherwise} \end{cases} \tag{95}$$

That $\lambda k.x \cdot f\,k$ is increasing if $x > \mathsf{zero}$ and f is increasing follows from mutually proving that $\cdot$ is monotone in the second argument, and that $\mathsf{zero} \cdot y = \mathsf{zero}$. Exponentiation x^y comes with similar caveats as multiplication, but works with the same strategy.

$$x^{\mathsf{zero}} \quad :\equiv \quad \mathsf{succ}\,\mathsf{zero} \tag{96}$$
$$x^{\mathsf{succ}\, y} \quad :\equiv \quad (x^y) \cdot x \tag{97}$$
$$x^{\mathsf{limit}\, f} \quad :\equiv \quad \begin{cases} \mathsf{zero} & \text{if } x = \mathsf{zero} \\ \mathsf{succ}\,\mathsf{zero} & \text{if } x = \mathsf{succ}\,\mathsf{zero} \\ \mathsf{limit}\,(\lambda k. x^{f\,k}) & \text{otherwise.} \end{cases} \tag{98}$$

With these definitions, the properties introduced in Section 4.3 are automatically satisfied, and our Agda formalisation shows that these properties describe the above operations uniquely:

Theorem 42 (⚙). Brw *has unique addition, multiplication, and exponentation, given by* (90) – (98). □

Many arithmetic properties can easily be established by induction, for example:

Lemma 43 (⚙).

(i) Addition and multiplication are weakly monotone in the first argument: if $x \leq y$ then $x + z \leq y + z$ and $x \cdot z \leq y \cdot z$.
(ii) Addition is left cancellative: if $x + y = x + z$ then $y = z$.

(iii) Addition and multiplication associative, and multiplication distributes over addition: $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$.

(iv) Exponentiation is a homomorphism: $x^{y+z} = x^y \cdot x^z$. □

The first infinite ordinal ω can be defined as a Brouwer tree as $\omega :\equiv \mathsf{limit}\,\iota$, where $\iota : \mathbb{N} \xrightarrow{\leq} \mathsf{Brw}$ embeds the natural numbers as finite Brouwer trees. As an example of how Brw behaves as expected as a type of ordinals, we can also define $\varepsilon_0 :\equiv \mathsf{limit}\,(\lambda k.\omega \uparrow\uparrow k)$, where $\omega \uparrow\uparrow (k+1) :\equiv \omega^{\omega \uparrow\uparrow k}$, and use the bisim constructor to show that indeed $\omega^{\varepsilon_0} = \varepsilon_0$. Similarly, using the lemmas we have already established, it is now not so hard to establish other expected properties of Brouwer trees, that would not hold without the path constructors in the definition of Brw:

Lemma 44 (⚙). *Brouwer trees of the form ω^x are additive principal:*

(i) if $a < \omega^x$ then $a + \omega^x = \omega^x$.

(ii) if $a < \omega^x$ and $b < \omega^x$ then $a + b < \omega^x$.

Furthermore, if $x >$ zero and $n : \mathbb{N}$, then $\iota(n + 1) \cdot \omega^x = \omega^x$.

Proof. For (i), by antisymmetry it is enough to show $a+\omega^x \leq \omega^x$, since $\omega^x \leq a+\omega^x$ holds by $\omega^x = 0 + \omega^x$ and monotonicity of addition. We prove $a + \omega^x \leq \omega^x$ by induction on a, making crucial use of Lemma 32. Statement (ii) is an easy corollary of (i) and strict monotonicity of $+$ in the second argument. The final statement is proven by induction on x, with an inner induction on n for the successor case. □

Perhaps more surprisingly, even though Brw has addition with expected properties, it is a constructive taboo that Brw has subtraction. This is in contrast with the situation for Cnf, where subtraction is computable, and in fact was crucial in our proof of correctness of the arithmentic operations.

Theorem 45 (⚙). *If* Brw *has subtraction, then it has unique subtraction.* Brw *has subtraction if and only if* LPO *holds.*

Proof. Firstly, note that the type of having subtraction for Brw is a proposition due to left cancellability of addition: if z and z' are candidates for $y - x$, then $x + z = y = x + z'$, hence $z = z'$ by Lemma 43(ii). Hence subtraction and unique subtraction coincide for Brw.

In Theorem 53, we will show that LPO holds if and only if $\mathsf{Splits}(\mathsf{Brw}, <, \leq)$. Thus it is sufficient to show that Brw has subtraction if and only if $\mathsf{Splits}(\mathsf{Brw}, <, \leq)$. If Brw has subtraction, then we can split $p : x \leq y$ by comparing $y -_p x$ with 0, which is possible by Lemma 30 — we have $x = y$ if $y -_p x = 0$, and $x < y$ if $y -_p x > 0$. Conversely, assume $\leq$ splits, and let x, y with $x \leq y$ be given. Since having subtraction for Brw is a proposition, we can use classifiability induction on y to construct $y -_p x$. In each case, we first split p: if $x = y$, then we define $y -_p x :\equiv$ zero. If instead $x < y$, we cannot have $y =$ zero, and if $y = \mathsf{succ}\,y'$, we can use Lemma 31 to define $y -_p x$ using the induction hypotheses. Finally if $y = \mathsf{limit}\,f$ we use Lemma 32 to again be able to use the induction hypothesis to finish the job. □

6.6. **Decidability and Undecidability for Brouwer Trees.** We now consider what is decidable and what is not for Brouwer trees. Because we can distinguish constructors by Lemma 30, we can decide most properties of finite Brouwer trees, i.e. Brouwer trees x with $x < \omega$.

Theorem 46 (⚙). *It is decidable whether a Brouwer tree is finite. If n is a finite Brouwer tree and $\sim$ is one of the relations $=$, $<$, $\leq$, then the predicates $(n \sim \_)$ and $(\_ \sim n)$ are decidable.*

Proof. Deciding finiteness is easy to do by induction, since $\mathsf{limit}\, f$ is never finite, and hence the bisim constructor is trivially respected. Similar considerations apply when deciding $(n \sim \_)$ and $(\_ \sim n)$; see the Agda formalisation for details. ◻

Theorem 46 covers the most important properties that are decidable. In order to demonstrate that certain other properties cannot be shown to be decidable, we use constructive taboos as discussed in Section 2.2. Many constructive taboos talk about binary sequences, while the sequences that are important in the construction of Brw are strictly increasing sequences of Brouwer trees. In order connect the taboos with properties of Brw, we therefore want to be able to translate between both types of sequences. Recall the construction of the *jumping sequence* in the proof of Theorem 17. For Brw, we can implement it as a concrete function of type

$$\cdot^{\uparrow} : (\mathbb{N} \to \mathbf{2}) \to (\mathbb{N} \xrightarrow{<} \mathsf{Brw}). \tag{99}$$

We then have:

Lemma 47 (). *For any binary sequence $s : \mathbb{N} \to \mathbf{2}$, we have $\mathsf{limit}\, s^{\uparrow} \leq \omega \cdot 2$. Moreover, the following three statements are equivalent:*

(i) $\exists k. s_k = \mathsf{tt}$
(ii) $\mathsf{limit}\, s^{\uparrow} = \omega \cdot 2$
(iii) $\omega < \mathsf{limit}\, s^{\uparrow}$

Proof. For any i, we have $s^{\uparrow}_i \leq \omega + i$, and thus $\mathsf{limit}\, s^{\uparrow} \leq \omega \cdot 2$. To see the equivalences, we have:

(i) ⇒ (ii): From (i), we can compute a minimal k such that $s_k = \mathsf{tt}$. Then, we have $s^{\uparrow}_i = i$ for $i < k$, and $s^{\uparrow}_{k+i} = \omega + i$. Therefore, $\mathsf{limit}\, s^{\uparrow} = \omega \cdot 2$.

(ii) ⇒ (iii): This is immediate as $\omega < \omega \cdot 2$.

(iii) ⇒ (i): By Lemma 32, there exists some n such that $\omega < s^{\uparrow}\, n$. In particular, $s^{\uparrow}\, n$ is infinite. By Lemma 2 and Theorem 46, we can therefore find an n such that $s^{\uparrow}\, n$ is infinite. The minimal such n has the property that $s_n = \mathsf{tt}$. ◻

Vice versa, assume $f : \mathbb{N} \to \mathsf{Brw}$ is a sequence and $P : \mathsf{Brw} \to \mathsf{hProp}$ a predicate such that for all n, $P(f\, n)$ is decidable. We then define the *unjumping sequence*

$$f^{\downarrow P} : \mathbb{N} \to \mathbf{2} \tag{100}$$

by setting

$$f^{\downarrow P}\, n :\equiv \begin{cases} \mathsf{tt} & \text{if } P(f\, n) \\ \mathsf{ff} & \text{if } \neg P(f\, n). \end{cases} \tag{101}$$

We now put the jumping sequence to work to show that most decidability questions of arbitrary Brouwer trees are in fact equivalent to each other, and to the constructive taboo of LPO.

Theorem 48 (). *For the type of Brouwer trees, the following statements are equivalent:*

(i) LPO
(ii) $\forall x, y. \mathsf{Dec}(x \leq y)$
(iii) $\forall x, y. \mathsf{Dec}(x < y)$
(iv) $\forall x. \mathsf{Dec}(\omega < x)$
(v) $\forall x, y. \mathsf{Dec}(x = y)$
(vi) $\forall x. \mathsf{Dec}(x = \omega \cdot 2)$

Proof. We show the equivalence using two cycles: (i) ⇒ (ii) ⇒ (iii) ⇒ (iv) ⇒ (i) and (i) ⇒ (v) ⇒ (vi) ⇒ (i).

(i) ⇒ (ii): Assume LPO and define $P(x) :\equiv \forall y. \mathsf{Dec}(x \leq y)$. We show $\forall x. P(x)$ by classifiability induction.

- case $P(\mathsf{zero})$: Trivial, since $\mathsf{zero} \leq y$ for any y.
- case $P(\mathsf{succ}\, x)$: To decide $\mathsf{succ}\, x \leq y$, we do classifiability induction on y. Using the results of Section 6.2, we know that $\mathsf{succ}\, x \leq \mathsf{zero}$ is false and that $\mathsf{succ}\, x \leq \mathsf{succ}\, y'$ holds iff $x \leq y'$, which is decidable by the induction hypothesis. Asked whether $\mathsf{succ}\, x \leq \mathsf{limit}\, f$, we use that $\mathsf{succ}\, x \leq f\, n$ is decidable by the induction hypothesis and consider the unjumping sequence $f^{\downarrow Q}$, with $Q(z) :\equiv (\mathsf{succ}\, x \leq z)$; i.e. we have
$$f^{\downarrow Q}(n) :\equiv \begin{cases} \mathsf{tt} & \textit{if } \mathsf{succ}\, x \leq f\, n \\ \mathsf{ff} & \textit{if } \neg(\mathsf{succ}\, x \leq f\, n). \end{cases} \tag{102}$$
Thanks to LPO, we know that the sequence $f^{\downarrow Q}$ is either constantly ff, or (cf. Lemma 2) we get n such that $\mathsf{succ}\, x \leq f\, n$. In the first case, we assume $\mathsf{succ}\, x \leq \mathsf{limit}\, f$; by Lemma 32, this implies $\exists n.\mathsf{succ}\, x \leq f\, n$, contradicting the statement that the sequence is constantly ff. In the second case, we clearly have $\mathsf{succ}\, x \leq \mathsf{limit}\, f$.
- case $P(\mathsf{limit}\, f)$: We have to decide $\mathsf{limit}\, f \leq y$. We use the unjumping sequence with $Q(z) :\equiv \neg(z \leq y)$, i.e.
$$f^{\downarrow Q}(n) :\equiv \begin{cases} \mathsf{tt} & \textit{if } \neg(f\, n \leq y) \\ \mathsf{ff} & \textit{if } f\, n \leq y \end{cases} \tag{103}$$
Again, we apply LPO. If the sequence is constantly ff, we have $\mathsf{limit}\, f \leq y$. If we get an n with $\neg(f\, n \leq y)$, then the assumption $\mathsf{limit}\, f \leq y$ gives a contradiction.

(ii) $\Rightarrow$ (iii): Trivial, since $(x < y) \equiv (\mathsf{succ}\, x \leq y)$.

(iii) $\Rightarrow$ (iv): The latter is a special case of the former.

(iv) $\Rightarrow$ (i): Assume (iv) and let s be a binary sequence. By assumption, we can decide $\omega < \mathsf{limit}\, s^{\uparrow}$, and thus $\exists i.s_i = \mathsf{tt}$ by Lemma 47, implying LPO.

(i) $\Rightarrow$ (v): Since $(x = y) \leftrightarrow (x \leq y) \wedge (y \leq x)$ by antisymmetry, this follows from the result (i) $\Rightarrow$ (ii) above.

(v) $\Rightarrow$ (vi): The latter is a special case of the former.

(vi) $\Rightarrow$ (i): Similar to the direction (iv) $\Rightarrow$ (i) above, this follows from Lemma 47. □

It is worth noting that the argument of (iv) $\Rightarrow$ (i) in the above proof shows LPO, while Theorem 17, under similar assumptions and with a similar strategy, only shows WLPO. The construction of a concrete n is made possible by the earlier results on Brw, but is not possible for the abstract situation considered by Theorem 17.

As we have just seen, being able to check equality with $\omega \cdot 2$ is equivalent to LPO, and equality with a finite number is always decidable. Equality with ω lies in-between, in the sense that it is equivalent to WLPO:

Theorem 49 (⚙). *Brw has locally decidable equality at ω if and only if WLPO holds:*
$$\mathsf{WLPO} \;\leftrightarrow\; \forall(x : \mathsf{Brw}).\mathsf{Dec}(x = \omega) \tag{104}$$

Proof. Assume $\forall x.\mathsf{Dec}(x = \omega)$. Let s be a binary sequence. If $\mathsf{limit}\, s^{\uparrow} = \omega$, then s is constantly ff; otherwise, s is not constantly ff.

Assume now WLPO and let $x : \mathsf{Brw}$ be given. If x is zero or a successor, then $x \neq \omega$; thus, we assume that x is $\mathsf{limit}\, f$. Consider $f^{\downarrow \neg \mathsf{isFinite}}$. If this sequence is constantly ff, then every f_i is finite and the limit is ω. If is it not constantly ff, then x must differ from ω. □

For comparison, we observe the following:

Theorem 50 (⚙). *Let n be a natural number larger or equal to 2. Deciding equalities locally at $\omega \cdot n$ is equivalent to* LPO*:*

$$\mathsf{LPO} \leftrightarrow \forall(x : \mathsf{Brw}).\mathsf{Dec}(x = \omega \cdot n). \tag{105}$$

Proof. By left cancellation of addition (Lemma 43), we have

$$(\forall x.\mathsf{Dec}(x = \omega + a)) \to (\forall x.\mathsf{Dec}(x = a)) \tag{106}$$

for any a. As a consequence, decidability of equality with $\omega \cdot n$, for $n \geq 2$, implies decidability of equality with $\omega \cdot 2$, which implies LPO, which implies decidability of equality by Theorem 48. □

Summarising Theorems 46, 49 and 50, we have shown that decidability of equality with $\omega \cdot n$ holds for $n = 0$, corresponds to WLPO for $n = 1$, and to LPO if $n \geq 2$. For stability, which is somewhat weaker than decidability, we have:

Theorem 51 (⚙). *Equality of* Brw *is stable at ω, but stability at $\omega \cdot n$ for $n \geq 2$ implies* MP*:*

$$\forall(x : \mathsf{Brw}).\mathsf{Stable}(x = \omega) \tag{107}$$

$$(\forall(x : \mathsf{Brw}).\mathsf{Stable}(x = \omega \cdot n)) \to \mathsf{MP}. \tag{108}$$

Proof. Assume $\neg\neg(x = \omega)$; then, since x cannot be zero or a successor, let $x = \mathsf{limit}\, f$. We have $x = \omega$ if and only if every f_i is finite. If any f_i is not finite, then $x \neq \omega$, in contradiction to the assumption.

For the second part, setting $x :\equiv \mathsf{limit}\, s^{\uparrow}$ and applying Lemma 47 shows immediately that the statement $\forall(x : \mathsf{Brw}).\mathsf{Stable}(x = \omega \cdot 2)$ implies MP. The general case for $n > 2$ again follows by left cancellation of addition (Lemma 43), since it implies

$$(\forall x.\mathsf{Stable}(x = \omega + a)) \to (\forall x.\mathsf{Stable}(x = a)) \tag{109}$$

for any a. □

Adding a finite number k does not change the situation in Theorems 49 to 51: If we replace $\omega \cdot n$ by $\omega \cdot n + k$, the remaining statements hold without any further difference.

The following observation will be the key ingredient of a proof that LPO implies trichotomy:

Lemma 52 (⚙). *If* LPO *holds then, for all x, y :* Brw*, we have $\neg(x \leq y) \to (y < x)$.*

Proof. Assume LPO. Note that by Lemma 1, we then also have MP. We do classifiability induction on x.

- Case $x \equiv \mathsf{zero}$: The assumption $\neg(\mathsf{zero} \leq y)$ is absurd.
- Case $x \equiv \mathsf{succ}\, x'$: We have to show $\neg(\mathsf{succ}\, x' \leq y) \to (y < \mathsf{succ}\, x')$. We do classifiability induction on y. The case $y \equiv \mathsf{zero}$ is trivial, and the case $y \equiv \mathsf{succ}\, y'$ follows from Lemma 31. Finally, we need to consider the situation $y \equiv \mathsf{limit}\, g$:

$$\begin{aligned} \neg(\mathsf{succ}\, x' \leq \mathsf{limit}\, g) \quad &\Rightarrow \quad \neg\exists i.\mathsf{succ}\, x' \leq g_i \\ &\Rightarrow \quad \forall i.\neg(\mathsf{succ}\, x' \leq g_i) \\ &\Rightarrow \quad \forall i.g_i < \mathsf{succ}\, x' \\ &\Rightarrow \quad \forall i.g_i \leq x' \\ &\Rightarrow \quad \mathsf{limit}\, g \leq x' \\ &\Rightarrow \quad \mathsf{limit}\, g < \mathsf{succ}\, x'. \end{aligned} \tag{110}$$

- The final case is $x \equiv \mathsf{limit}\, f$:

$$\begin{aligned}
\neg(\mathsf{limit}\, f \leq y) \quad &\Rightarrow \quad \neg(\forall i. f_i \leq y) \\
&\Rightarrow \quad \neg\neg(\exists i. \neg(f_i \leq y)) \\
&\Rightarrow \quad \neg\neg(\exists i. y < f_i) \\
&\qquad \textit{(using } \mathsf{MP} \textit{ on a decidable family of propositions)} \\
&\Rightarrow \quad \exists i. y < f_i \\
&\Rightarrow \quad y < \mathsf{limit}\, f.
\end{aligned} \tag{111}$$

□

Using this lemma, and going via LPO, we can now show that splitting $\leq$ implies trichotomy for Brw — in the general setting, only the reverse direction, i.e., that trichotomy implies splitting, holds.

Theorem 53 (⚙). *The following properties are equivalent for the type of Brouwer trees:*

(i) LPO
(ii) trichotomy: $\forall x, y.(x < y) \uplus (x = y) \uplus (y < x)$
(iii) splitting: $\forall x, y.(x \leq y) \to (x < y) \uplus (x = y)$.

Proof. We show (i) $\Rightarrow$ (ii) $\Rightarrow$ (iii) $\Rightarrow$ (i).

(i) $\Rightarrow$ (ii): Assume LPO. By Theorem 48, we can decide $x < y$. If it holds, we are done. Otherwise, by the contrapositive of Lemma 52, we get $\neg\neg(y \leq x)$ and, since decidability implies stability, therefore $y \leq x$. In the same way, we decide $y < x$, which gives us either $y < x$ or $x \leq y$. The second case implies $x = y$ by antisymmetry.

(ii) $\Rightarrow$ (iii): This is an instance of Lemma 16.

(iii) $\Rightarrow$ (i): Given a binary sequence s, we know $\mathsf{limit}\, s^{\uparrow} \leq \omega \cdot 2$ from Lemma 47. Splitting this inequality allows us to decide whether $\mathsf{limit}\, s^{\uparrow} = \omega \cdot 2$ which, again by Lemma 47, yields the conclusion of LPO. □

Another natural question is if we can compute suprema of not necessarily increasing sequences of Brouwer trees. An important special case is the join or maximum $x \sqcup y$ of two trees, which is the suprema of the sequence $(x, y, y, y, \ldots)$. This is easy to compute if one of the trees is at most ω, as $\omega \leq \mathsf{limit}\, f$ for any $f : \mathbb{N} \xrightarrow{<} \mathsf{Brw}$:

Theorem 54 (⚙). *If $y = n$ for a finite n, or $y = \omega$, we can define a function* $(\_ \sqcup y) : \mathsf{Brw} \to \mathsf{Brw}$ *calculating the binary join with y.*

Proof. For $y = n$ finite, we can define

$$x \sqcup n = \begin{cases} x & \text{if } n = 0 \text{ or } x = \mathsf{limit}\, f \\ n & \text{if } x = 0 \\ \mathsf{succ}\,(x' \sqcup n') & \text{if } x = \mathsf{succ}\, x' \text{ and } n = \mathsf{succ}\, n' \end{cases} \tag{112}$$

and prove that this indeed is the join of x and n. Indeed any limit is going to be larger than any finite n, and 0 is always the smallest element, hence $x \sqcup 0 = 0 \sqcup x = x$. If both Brouwer trees are successors, we know that their join is a successor as well.

For joins with $y = \omega$, note that by Theorem 46, we can decide if x is finite or not. Clearly a finite x is smaller than ω, and ω is the smallest infinite Brouwer tree, leading to the following definition:

$$x \sqcup \omega = \begin{cases} \omega & \text{if } x \text{ is finite} \\ x & \text{otherwise} \end{cases} \tag{113}$$

See our Agda formalisation for the proof that this indeed is the join of x and ω. □

This is as far as we can go — already for $y = \omega + 1$ being able to calculate $x \sqcup y$ would imply a constructive taboo:

Theorem 55 (⚙). *If* LPO *holds, then* $x \sqcup (\omega + 1)$ *exists for every* $x : \mathsf{Brw}$*. If* $x \sqcup (\omega + 1)$ *exists for every* $x : \mathsf{Brw}$*, then* WLPO *follows.*

Proof. Assume LPO and let $x : \mathsf{Brw}$ be given. By Theorem 48, we can decide $\omega < x$. If this is the case, it is easy to see that $x = x \sqcup (\omega + 1)$. If it is not the case and $x \equiv \mathsf{limit}\, f$, then every f_i must be finite, implying $x = \omega$, and the binary join is $\omega + 1$. If $\neg(\omega < x)$ and x is a successor, then x must be finite, again allowing us to see that the join is $\omega + 1$.

Now, assume that $x \sqcup (\omega + 1)$ exists for any x. We show that we can decide the equality $x = \omega$ which, by Theorem 49, implies WLPO. When deciding the proposition $x = \omega$, we can assume $x \equiv \mathsf{limit}\, f$, as the other cases are trivial. Using Theorem 41, we can check whether $(\mathsf{limit}\, f) \sqcup (\omega + 1)$ is a successor or a limit. In the first case, any f_i being infinite would lead to a contradiction, thus every f_i must be finite, and $\mathsf{limit}\, f = \omega$ follows. In the second case, we observe that $\mathsf{limit}\, f = \omega$ would imply that the join with $\omega + 1$ is $\omega + 1$, yielding a contradiction. □

6.7. **An Alternative Equivalent Definition of Brouwer Trees.** We also considered an alternative quotient inductive-inductive construction of Brouwer trees using a path constructor

$$\mathsf{antisym} : \ x \leq y \to y \leq x \to x = y \tag{114}$$

following the construction of the partiality monad [3]. This constructor should be seen as a more powerful version of the bisim constructor, since if $f \lesssim g$, then $\mathsf{limit}\, f \leq \mathsf{limit}\, g$. By Theorem 7.2.2 of the HoTT book [64, Thm 7.2.2], this constructor further implies that the constructed type is a set. Let us write Brw' for the variation of Brw which uses the constructor (114) instead of bisim.

Of course, Brw' has antisymmetry for free, but the price to pay is that the already very involved proof of well-foundedness becomes significantly more difficult. After proving antisymmetry for Brw, we managed to prove $\mathsf{Brw} \simeq \mathsf{Brw}'$, thus establishing well-foundedness for Brw' — but we did not manage to prove this directly.

Note that the constructor limit of Brw asks for strictly increasing sequences; without that condition, extensionality fails. For Brw', one can consider removing the condition, but Brw' is then no longer equivalent to Brw. Most importantly, the constructors overlap and the ability to decide whether an element is zero is lost, without which we do not know how to define e.g. exponentiation on Brw.

7. Transitive, Extensional and Well-Founded Orders

As introduced in Section 3.3, Ord is the type of ordinals $(X, \prec)$ where the order is well-founded, extensional, and transitive. As noticed by Escardó [33], extensionality and Lemma 3.3 of Kraus, Escardó, Coquand and Altenkirch [46, Lem 3.3] imply that X is a set. It further follows that also Ord is a set [64, Thm 10.3.10].

Clearly, the identity function is a simulation, and the composition of two (bounded) simulations is a (bounded) simulation; thus, $\leq$ is reflexive and $\leq$ as well as $<$ are transitive. Note that the simulation requirement (b) is a proposition by Corollary 10.3.13 of the HoTT book [64, Cor 10.3.13] (i.e. even if formulated using Σ rather than $\exists$), and $X \leq Y$ is a proposition [64, Lem 10.3.16]. As a consequence, $\leq$ is antisymmetric. Similarly, if a simulation is bounded, then the bound is unique, and hence also the type $X < Y$ is a proposition.

We recall a result of the HoTT book that we will use to prove non-constructive results:

Lemma 56 ([64, Thm 10.4.3])**.** *Assuming* LEM*,* $(A, \prec)$ *is an ordinal if and only if every nonempty subset of* A *has a least element.* □

Given ordinals A and B, one can construct a new ordinal $A \uplus B$, reusing the order on each component, and letting $\mathsf{inl}(a) \prec_{A \uplus B} \mathsf{inr}(b)$. This construction has been formalised by Escardó [33], and amounts to the *categorical join*. This is used in the proof of the second part of the following lemma:

Lemma 57. *If* $A < B$ *and* $B \leq C$ *then* $A < C$*. However* $A \leq B$ *and* $B < C$ *implies* $A < C$ *if and only if excluded middle* LEM *holds.*

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 Lemma 56 and $A \simeq C_{/c}$. Conversely, let P be a proposition; it is an ordinal with the empty order. Consider also the unit type $\mathbf{1}$ as 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. □

As noted in the HoTT book [64, Thm 10.3.20], Ord itself carries the structure of an extensional well-founded order, and so is an element of Ord, albeit in the next higher universe.

Theorem 58. *The order* $<$ *on* Ord *is well-founded, extensional, and transitive.* □

Since $<$ is well-founded, it is also irreflexive by Lemma 4.

Corollary 59. *The triple* $(\mathsf{Ord}, <, \leq)$ *satisfies the assumptions (A1), (A2), and (A3).*

Proof. Most requirements are given by Theorem 58, which in particular implies that $<$ is irreflexive. Lemma 57 shows (A3). The remaining properties follow directly from the definitions. □

There is exactly one order on the empty type $\mathbf{0}$, and this order makes $\mathbf{0}$ into a zero for Ord. Similarly there is only one irreflexive order on the unit type $\mathbf{1}$, namely the one where the only element is not related to itself. The successor of A is given by A adjoined with $\mathbf{1}$ to the right $A \uplus \mathbf{1}$, thus adding one more element greater than all the given elements. As proven by de Jong and Escardó [25], notably Ord has suprema of arbitrary small families of ordinals, not only of $\mathbb{N}$-indexed families, or of strictly increasing sequences, such as the case for Brw.

Lemma 60 (⚙)**.** *The type* $\mathbf{0}$ *is zero. The strong successor of* A *is* $A \uplus \mathbf{1}$*, and if* $F : X \to \mathsf{Ord}$ *is an* X*-indexed family of ordinals, then its supremum* $\mathsf{sup}\, F$ *is the quotient* $(\Sigma x : X.Fx)/\sim$*, where* $(x, y) \sim (x', y')$ *if and only if* $(Fx)_{/y} \simeq (Fx')_{/y'}$*, with* $[(x, y)] \prec [(x', y')]$ *if* $(Fx)_{/y} < (Fx')_{/y'}$*.*

Proof. Zero is clear. The definition of a bounded simulation implies $(X < Y) \leftrightarrow (X \uplus \mathbf{1} \leq Y)$, making Lemma 6 applicable. The definition of the type $\mathsf{sup}\, F$ can be found in the HoTT book [64, Lem 10.3.22], and the proof that it is indeed the supremum was given by de Jong and Escardó [25, Thm 5.12]. □

Theorem 61. 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 63 below). This allows us to use the explicit representation of suprema from Lemma 60 in the limit cases. □

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

Theorem 62. Ord *has subtraction if and only if* LEM *holds.*

Proof. Let P be a proposition and assume that Ord has subtraction. Then, there is Q such that $P \uplus Q = \mathbf{1}$. This implies $Q \leftrightarrow \neg P$, and the assumed equation becomes $P \uplus \neg P$.

For the other direction, assume LEM and let $s : X \leq Y$ be given. Defining $X_1 :\equiv \Sigma(y : Y).\neg s^{-1}(y)$ ensures $X \uplus X_1 = Y$, where LEM is required to show that the canonical function is a simulation (equivalently, to construct the inverse). □

Theorem 63. *Each of the following statements on its own implies the law of excluded middle (*LEM*), and each of the first five statements is equivalent to* LEM*:*

(i) The successor $(\_ \uplus \mathbf{1})$ *is* $\leq$*-monotone.*
(ii) The successor $(\_ \uplus \mathbf{1})$ *is* $<$*-monotone.*
(iii) $<$ *is trichotomous, i.e.* $(X < Y) \uplus (X = Y) \uplus (X > Y)$.
(iv) $\leq$ *is connex, i.e.* $(X \leq Y) \uplus (X \geq Y)$.
(v) Ord *has weak classification.*
(vi) Ord *has classification.*
(vii) Ord *satisfies classifiability induction.*

Proof. We first show the chain LEM $\Rightarrow$ (i) $\Rightarrow$ (ii) $\Rightarrow$ LEM.

LEM $\Rightarrow$ (i): Let $f : A \leq B$. Using LEM, there is a minimal $b : B \uplus \mathbf{1}$ which is not in the image of f by Lemma 56. The simulation $A \uplus \mathbf{1} \leq B \uplus \mathbf{1}$ is given by $f \uplus b$.

(i) $\Rightarrow$ (ii): Assume we have $A < B$. By Lemmas 6 and 60, this is equivalent to $A \uplus \mathbf{1} \leq B$. Assuming $(\_ \uplus \mathbf{1})$ is $\leq$-monotone, we get $A \uplus \mathbf{2} \leq B \uplus \mathbf{1}$, and applying Lemma 6 once more, this is equivalent to $A \uplus \mathbf{1} < B \uplus \mathbf{1}$.

(ii) $\Rightarrow$ LEM: Assume P is a proposition. We have $\mathbf{0} < \mathbf{1} \uplus P$. If $(\_ \uplus \mathbf{1})$ 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 LEM $\Rightarrow$ (iii) $\Rightarrow$ (iv) $\Rightarrow$ LEM, where the first implication is given by Theorem 10.4.1 of the HoTT book [64, Thm 10.4.1].

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

(iv) $\Rightarrow$ LEM: Given a proposition P, 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.

The next step is to show (vii) $\Rightarrow$ (vi) $\Rightarrow$ (v) $\Rightarrow$ LEM. The first implication is Corollary 10 and the second implication is automatic since any classifiable ordinal is also weakly classifiable. We have (v) $\Rightarrow$ LEM because a classifiable proposition P is either $\mathbf{0}$ (thus $\neg P$) or a successor $X \uplus \mathbf{1}$ (thus P and necessarily $X = \mathbf{0}$) or the supremum of $\mathsf{fst} : \mathsf{Ord}_{/X} \to \mathsf{Ord}$ where there exists $X_0 < P$ (this case is impossible).

Finally, we check LEM $\Rightarrow$ (v). Thus, we assume LEM and a given X. If X is the supremum of $\mathsf{fst} : \mathsf{Ord}_{/X} \to \mathsf{Ord}$ then either X is empty or X is a general limit and we are done.

Now assume that X is not the supremum of $\mathsf{fst} : \mathsf{Ord}_{/X} \to \mathsf{Ord}$. Since X certainly satisfies the first part of the definition (36) and we have LEM (as well as Lemma 56) at our disposal, this means that there is some Y with

$$\forall X'.(X' < X) \to (X' \leq Y) \tag{115}$$

together with $\neg(X \leq Y)$ which, by the previous parts of this theorem, implies $Y < X$. With the terminology of Section 4.2.1, this means that Y is a predecessor of X. We show that X in fact is the strong successor of Y. To do so, we additionally need that, for a given Y' such that $Y < Y'$, we have $X \leq Y'$. Assume not: since

$\mathsf{LEM} \Rightarrow$ (iii), we then have $Y' < X$, and using (115) therefore $Y' \leq Y$, leading to the contradiction $Y < Y' \leq Y$. □

Last but not least, we consider splitting:

Theorem 64. *Inequalities in* Ord *split* $((X \leq Y) \to (X = Y) \uplus (X < Y))$ *if and only if* LEM *holds.*

Proof. For the "only if" direction, let P be a proposition; we always have $P \leq \mathbf{1}$. If we can split this inequality, it follows that $P \uplus \neg P$.

For the other direction, we assume LEM and $e : X \leq Y$. If e is surjective, then it is an equivalence and $X = Y$ follows. If e is not surjective, the complement of its image has a smallest element y_0 by Lemma 56. Thus, e is bounded by y_0 and witnesses $X < Y$. □

8. Interpretations Between the Notions

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

8.1. 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:

$$\mathsf{CtoB}(0) :\equiv \mathsf{zero} \tag{116}$$

$$\mathsf{CtoB}(\omega^a + b) :\equiv \omega^{\mathsf{CtoB}(a)} + \mathsf{CtoB}(b) \tag{117}$$

Theorem 65 (). *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 $\omega^a + b < \omega^c + d$ because $a < c$ uses Lemma 44.

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

We remark once again that the above proof was only possible because of the "correct" definition of Brw — it would not be the case that CtoB preserves $<$ if we had used a "naive" version of Brouwer trees without path constructors. By reflecting $\leq$ and antisymmetry, we have:

Corollary 66 (). *The function* CtoB *is injective.*

Proof. $\mathsf{CtoB}(a) = \mathsf{CtoB}(b)$ implies $\mathsf{CtoB}(a) \leq \mathsf{CtoB}(b)$ and thus, by Theorem 65, $a \leq b$. Analogously, one has $b \leq a$. Antisymmetry gives $a = b$. □

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 (Lemma 44) — see our formalisation for details.

Theorem 67 (). 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 $a = \omega^x + u$ and $b = \omega^y + v$. If $x < y$, then $a + b = b$. We have also $\omega^x < \omega^y$, which implies $\omega^{\mathsf{CtoB}(x)} < \omega^{\mathsf{CtoB}(y)}$ by Theorem 65. Then by Lemma 44(i) we have $\omega^{\mathsf{CtoB}(x)} + \omega^{\mathsf{CtoB}(y)} = \omega^{\mathsf{CtoB}(y)}$. By the same argument, from the fact $u < \omega^y$ 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 $a + b = \boldsymbol{\omega}^x \mathbin{\boldsymbol{+}} u + b$ by definition. By the 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 $\omega^{\mathsf{CtoB}(x)} + \mathsf{CtoB}(u) + \mathsf{CtoB}(b)$. □

Although we cannot calculate suprema in Cnf, we can still ask whether CtoB preserves those that exist. We restrict the question to the case of strictly increasing sequences. We first note that CtoB does preserve the *fundamental sequences* that we constructed for each limit CNF in the proof of Lemma 26, in the sense that CtoB sends x, the limit of its fundamental sequence s, to the limit of $\mathsf{CtoB} \circ s$.

Lemma 68 (). *Let $x : \mathsf{Cnf}$ be a limit, and s its fundamental sequence as assigned in the proof of Lemma 26. We then have $\mathsf{CtoB}(x) = \mathsf{limit}(\mathsf{CtoB} \circ s)$.*

Proof. Let x be given. We analyse how the fundamental sequence s of x is constructed and compute, using Theorem 67 extensively. In the following, we leave the embedding of natural numbers into both Cnf and Brw implicit, for convenience.

(i) Case $x \equiv \boldsymbol{\omega}^{c+1} \mathbin{\boldsymbol{+}} 0$: By definition, we have

$$\begin{aligned} \mathsf{CtoB}(\boldsymbol{\omega}^{c+1} \mathbin{\boldsymbol{+}} 0) &= \omega^{\mathsf{CtoB}(c+1)} \\ &= \omega^{\mathsf{CtoB}(c)+1} \\ &= \omega^{\mathsf{CtoB}(c)} \cdot \omega \\ &= \omega^{\mathsf{CtoB}(c)} \cdot \mathsf{limit}(\lambda i.i) \\ &= \mathsf{limit}(\omega^{\mathsf{CtoB}(c)} \cdot i) \end{aligned} \tag{118}$$

On the other hand, the fundamental sequence is in this case defined as $s :\equiv \lambda i.(\boldsymbol{\omega}^c \mathbin{\boldsymbol{+}} 0) \cdot i)$, and therefore

$$\begin{aligned} \mathsf{limit}(\mathsf{CtoB} \circ s) &= \mathsf{limit}\,(\lambda i.\mathsf{CtoB}((\boldsymbol{\omega}^c \mathbin{\boldsymbol{+}} 0) \cdot i)) \\ &= \mathsf{limit}\,(\lambda i.\mathsf{CtoB}(\boldsymbol{\omega}^c \mathbin{\boldsymbol{+}} 0) \cdot i) \\ &= \mathsf{limit}(\lambda i.\omega^{\mathsf{CtoB}(c)} \cdot i) \end{aligned} \tag{119}$$

(ii) Case $x \equiv \boldsymbol{\omega}^a \mathbin{\boldsymbol{+}} 0$, where a is not a successor: Writing r for the fundamental sequence of a, then the fundamental sequence of s is, by definition, $\lambda i.\boldsymbol{\omega}^{r_i} \mathbin{\boldsymbol{+}} 0$. Using that CtoB preserves the limit of r by induction, we have

$$\begin{aligned} \mathsf{CtoB}(\boldsymbol{\omega}^a \mathbin{\boldsymbol{+}} 0) &= \omega^{\mathsf{CtoB}(a)} \\ &= \omega^{\mathsf{limit}(\mathsf{CtoB} \circ r)} \\ &= \mathsf{limit}(\lambda i.\omega^{\mathsf{CtoB}(r_i)}) \\ &= \mathsf{limit}(\lambda i.\mathsf{CtoB}(\omega^{r_i})) \\ &= \mathsf{limit}(\mathsf{CtoB} \circ s). \end{aligned} \tag{120}$$

(iii) If $x = \boldsymbol{\omega}^a \mathbin{\boldsymbol{+}} b$ with $b > 0$, then b necessarily is a limit with fundamental sequence r. Recall that the fundamental sequence of x is then given by $\lambda i.\boldsymbol{\omega}^a \mathbin{\boldsymbol{+}} r_i$.

$$\begin{aligned} \mathsf{CtoB}(\boldsymbol{\omega}^a \mathbin{\boldsymbol{+}} b) &= \omega^{\mathsf{CtoB}\,a} + \mathsf{CtoB}\,b \\ &= \omega^{\mathsf{CtoB}\,a} + \mathsf{limit}(\mathsf{CtoB} \circ r) \\ &= \mathsf{limit}(\lambda i.\omega^{\mathsf{CtoB}\,a} + \mathsf{CtoB}(r_i)) \\ &= \mathsf{limit}(\lambda i.\mathsf{CtoB}(\boldsymbol{\omega}^a \mathbin{\boldsymbol{+}} r_i)) \\ &= \mathsf{limit}(\mathsf{CtoB} \circ s). \end{aligned} \tag{121}$$

□

This might seem like an encouraging first step, but in fact continuity of CtoB in general turns out to be a constructive taboo. This is because Cnf and Brw are powerful in different ways: if CtoB were to preserve limits, then we could use the decidable equality of Cnf to confirm that a CNF is the limit of some sequence, then transfer the limit across to Brw where we could use the strong property of strict inequalities below limits factoring through one of the elements of the sequence to find an explicit witness. Using this idea, we can show that continuity of CtoB implies Markov's principle. Conversely, Markov's principle proves that CtoB is continuous with respect to strictly increasing sequences, so we have an exact correspondence.

Theorem 69 (⚙). CtoB *preserves limits of strictly increasing sequences if and only if* MP *holds.*

Proof. First, assume that CtoB preserves limits of strictly increasing sequences. We want to show MP. Therefore, let s be a binary sequence such that $\neg(\forall i.s_i = \mathsf{ff})$. We claim that $\omega + \omega$ is the limit of the jumping sequence $s^{\uparrow} : \mathbb{N} \to \mathsf{Cnf}$ from (59). The first condition we need to check for $\omega + \omega$ to be the limit is $\forall n.s^{\uparrow}\, n \leq \omega + \omega$; but this is easy since every $s^{\uparrow}\, n$ is either finite or of the form $\omega + k$, depending on the decidable property of whether there is $i \leq n$ with $s_i = \mathsf{tt}$. The second condition requires us to check $\forall c.(\forall n.s^{\uparrow}\, n \leq c) \to \omega + \omega \leq c$. If $c < \omega + \omega$, then each $s^{\uparrow}\, i$ is below ω and thus $s_i = \mathsf{ff}$, which contradicts the assumption. Therefore, we have $\omega + \omega \leq c$ thanks to the trichotomy of Cnf by Theorem 18.

By assumption, we thus have $\mathsf{limit}\,(\mathsf{CtoB} \circ s^{\uparrow}) = \omega + \omega$. By Lemma 32, there exists n such that $\mathsf{CtoB}(s^{\uparrow}\,(n+1)) > \omega$, and by Theorem 65 CtoB reflects this inequality, hence $s^{\uparrow}\,(n+1) > \omega$ (since CtoB preserves ω by Theorem 67). This means $s^{\uparrow}\,(n+1) = \omega + k$ for some k, and indeed we must have $s_{n-k} = \mathsf{tt}$, i.e., we have proven $\exists i.s_i = \mathsf{tt}$, as required.

For the other direction, we first show the following:

Claim: Assume we have $x, y : \mathsf{Cnf}$ and strictly increasing sequences $f, g : \mathbb{N} \to \mathsf{Cnf}$ such that x is-$\mathbb{N}$-lim-of f and y is-$\mathbb{N}$-lim-of g. If $x \leq y$ and MP holds, then f is simulated by g:

$$\forall i.\exists k.f_i \leq g_k. \tag{122}$$

To see this, let us fix i and define $h : \mathbb{N} \to \mathbf{2}$ by

$$h_k :\equiv \begin{cases} \mathsf{ff} & \textit{if } g_k < f_i \\ \mathsf{tt} & \textit{if } f_i \leq g_k. \end{cases} \tag{123}$$

If $\forall k.h_k = \mathsf{ff}$, then $\forall k.g_k < f_i$ and therefore $y \leq f_i < x$ in contradiction to the assumption. Therefore, using MP, we have $\exists k.h_k = \mathsf{tt}$, i.e. $\exists k.f_i \leq g_k$.

We are now ready to show that MP implies that CtoB preserves limits of strictly increasing sequences. Let f be such a sequence with limit x; we want to show $\mathsf{CtoB}(x) = \mathsf{limit}(\mathsf{CtoB} \circ f)$. Let s be the fundamental sequence of x as constructed in the proof of Lemma 26. Using the above claim and MP, we see that s and f are bisimilar. Since CtoB preserves the relations (Theorem 65), $\mathsf{CtoB} \circ f$ and $\mathsf{CtoB} \circ s$ are bisimilar in Brw and therefore have equal limits. Hence using Lemma 68, we have $\mathsf{CtoB}(x) = \mathsf{limit}(\mathsf{CtoB} \circ s) = \mathsf{limit}(\mathsf{CtoB} \circ f)$, as required. □

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}, $\omega^{\omega^{\omega}}$, ...

Theorem 70 (⚙). *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 44, strict monotonicity of ω^{-}, and the induction hypothesis. □

8.2. **From Brouwer Trees to Extensional Well-Founded Orders.** As Brw comes with an order that is well-founded, extensional, 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, well-founded orders. We define

$$\mathsf{BtoO}(a) = \Sigma(y : \mathsf{Brw}).(y < a). \tag{124}$$

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 $\mathsf{BtoO}(a) \leq \mathsf{Brw}$:

Lemma 71. *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 [64, Thm 10.3.16]. If the equality holds then, again, the two properties in the definition of a simulation are clear for f. □

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 72 (⚙). *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 Definition 10.3.19 in the HoTT book [64, Def 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 71, 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 71, thus BtoO preserves $\leq$. This implies that $<$ is preserved as well since $X < Y \leftrightarrow (X \uplus 1) \leq Y$. □

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 73. *Assuming* LEM*, 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 $b \leq \mathsf{BtoO}(x)$ by Lemma 56. □

We do not know whether the reverse of Theorem 73 is provable, but from the assumption that BtoO is a simulation, we can derive another constructive taboo:

Theorem 74. *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 $\mathbf{2} \equiv \mathsf{BtoO}(\mathsf{succ}(\mathsf{succ\,zero}))$ and using the assumption that BtoO is a simulation, we get $b < \mathsf{succ}(\mathsf{succ\,zero})$ such that $\mathsf{BtoO}(b) = \bigsqcup S$. It is decidable whether b is zero or $\mathsf{succ\,zero}$, and this determines if s is constantly ff or not. □

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. $\mathsf{BtoO}(x + y) \geq \mathsf{BtoO}(x) \uplus \mathsf{BtoO}(y)$ and $\mathsf{BtoO}(x \cdot y) \geq \mathsf{BtoO}(x) \times \mathsf{BtoO}(y)$.

9. Computational Efficiency of Our Notions of Ordinals

Apart from logical expressiveness, it is also interesting to compare the computational efficiency of our different notions of ordinal. This is possible to do, since we have formalised them in Cubical Agda, which has computational support for higher inductive types and the Univalence Axiom. Inspired by Berger’s benchmarking of ordinal recursive versus higher type programs extracted from Gentzen’s proof of transfinite induction up to ε_0 [6], we compared the efficiency of our different ordinal representations for computing $H_{\omega^n}(1)$, where, for each notion of ordinal $\mathcal{O}$, $H : \mathcal{O} \to \mathbb{N} \to \mathbb{N}$ is the Hardy hierarchy [67], with

$$H_0(n) = n \tag{125}$$
$$H_{\alpha+1}(n) = H_\alpha(n+1) \tag{126}$$
$$H_{\lim f}(n) = H_{f(n)}(n). \tag{127}$$

However, since obviously $H_{\lim f}$ depends on the choice of fundamental sequence f in the limit case, this is not a well defined function on ordinals. To work around this issue, we instead compute $H : \mathcal{O} \to \mathbb{N} \to \|\mathbb{N}\|$, where $\|\mathbb{N}\|$ is the propositional truncation of $\mathbb{N}$. All elements of $\|\mathbb{N}\|$ are propositionally equal, but we can still let Cubical Agda compute their normal form, which will be of the form $|k|$ for some numeral k, which we can extract for a closed program. We are thus interested in the following defining equations:

$$H_0(n) = |n| \tag{128}$$
$$H_{\alpha+1}(n) = H_\alpha(n+1) \tag{129}$$
$$H_{\lim f}(n) = H_{f(n)}(n) \tag{130}$$

For $\mathcal{O} = \mathsf{Brw}$, this definition can now be implemented directly by induction on the Brouwer tree, whereas for $\mathcal{O} = \mathsf{Cnf}$, we use classifiability induction to define H. We cannot define H at all for $\mathcal{O} = \mathsf{Ord}$, since classification for Ord is a constructive taboo. Using Cubical Agda’s `--erased-cubical` feature, we compiled these definitions and ran $H_{\omega^n}(1)$ for increasing values of n — the result is the same for each n, but the run time increases. The results can be found in Figure 1. As can be seen there, Cantor normal forms are significantly more efficient than Brouwer trees for this computation. This could be in part due to their first-order representation, but also perhaps due to our implementation of classifiability induction for Cantor normal forms: this follows Gentzen’s proof of transfinite induction up to ε_0, and as Berger [6] noticed, this gives rise to an efficient, higher-order implementation.

10. Conclusions and Future Directions

We have introduced Cantor normal forms (Cnf), Brouwer trees (Brw), and extensional well-founded orders (Ord), three different approaches to ordinal theory

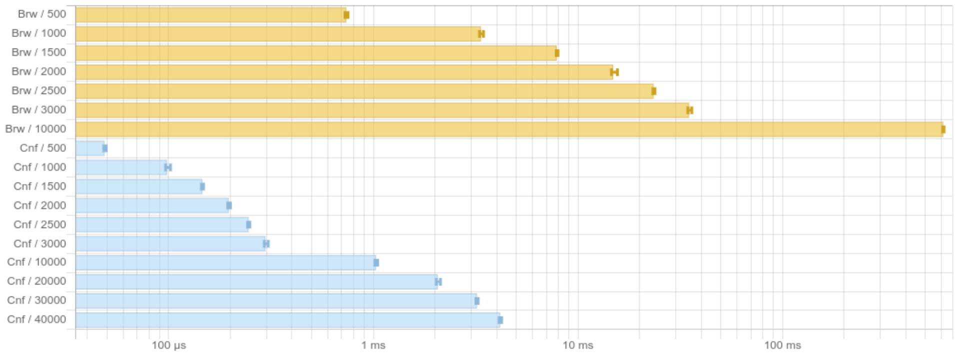

FIGURE 1. Benchmarking running time for $H_{\omega^n}(1)$, $n = 500$, 1000, . . . , for Brouwer trees (Brw) and Cantor normal forms (Cnf).

in the setting of homotopy type theory. Even though these approaches are quite different in their implementation, we have shown that they can all be studied in a single abstract setting and shown to share many expected properties of ordinals from their classical theory. It is our hope that our work may shed light on other constructive or formalised approaches to ordinals also in other settings [12, 13, 50, 54].

Cantor normal forms are a formulation where most properties are decidable, while the opposite is the case for extensional well-founded orders. Brouwer trees sit in the middle, with some of its properties being decidable, such as being a finite ordinal. However other properties, such as deciding equality between Brouwer trees in general, is a constructive taboo in the sense that it is equivalent to the non-constructive principle LPO. This is in contrast to the situation for extensional well-founded orders, where decidability of most properties is equivalent to the constructively much stronger principle LEM. In the future, we plan to investigate such decidability aspects further, including the notion of *semidecidability* [24] from synthetic computability theory [5, 36]. For example, if $c : \mathsf{Cnf}$ is smaller than ω^2, then the families $(\mathsf{CtoB}\, c \leq \_)$ and $(\mathsf{CtoB}\, c < \_)$ are semidecidable.

Along another dimension, the canonical maps $\mathsf{CtoB} : \mathsf{Cnf} \to \mathsf{Brw}$ and $\mathsf{BtoO} : \mathsf{Brw} \to \mathsf{Ord}$ embeds "smaller" types of ordinals into "larger" ones: while every element of Cnf represents an ordinal below ε_0, Brw can go much further, and since Brw can be viewed as an element of Ord, the latter can clearly reach larger ordinals than the former by the Burali-Forti argument [10, 18]. To at least partially overcome these limitations comparing Cnf to Brw, it would be interesting to consider more powerful ordinal notation systems such as those based on the Veblen functions [55, 65] or collapsing functions [4, 16], and see how these compare to Brouwer trees.

One can also explore more powerful variations of Brouwer trees. Following Schwichtenberg's approach [57], we could replace limits of countable sequences with larger limits and construct higher number classes as quotient inductive-inductive types in a similar way, e.g. a type Brw_2 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 well-foundedness 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 [31] and higher inductive types [49] can increase the proof-theoretical strength, and so, we hope that they can also be used to faithfully represent even larger ordinals.

Acknowledgements

We would like to thank the participants of the conferences MFCS'21, TYPES'21, TYPES'22, CCC'22, MGS'22, and PC'22 for their comments and suggestions related to this work. In particular, we are grateful to Tom de Jong, Helmut Schwichtenberg, Thorsten Altenkirch, Martín Hötzel Escardó, Peter Hancock, Paul Taylor, and Andreas Abel for insightful discussions. We also thank the anonymous reviewers of the current paper and its previous MFCS'21 version for their comments.

Funding: This work was supported by The Royal Society (grant reference URF\R1\191055) and the UK National Physical Laboratory Measurement Fellowship project "Dependent types for trustworthy tools".

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] Andrej Bauer. First steps in synthetic computability theory. *Electronic Notes in Theoretical Computer Science*, 155:5–31, 2006.

[6] Ulrich Berger. Program extraction from Gentzen's proof of transfinite induction up to ε_0. In R. Kahle, P. Schroeder-Heister, and R. Stärk, editors, *Proof Theory in Computer Science*, volume 2138 of *LNCS*, pages 68–77. Springer, 2001.

[7] Ulrich Berger. Realisability for induction and coinduction with applications to constructive analysis. *J. Univers. Comput. Sci.*, 16(18):2535–2555, 2010.

[8] Ulrich Berger and Monika Seisenberger. Proofs, programs, processes. *Theory Comput. Syst.*, 51(3):313–329, 2012.

[9] Ulrich Berger and Hideki Tsuiki. Intuitionistic fixed point logic. *Ann. Pure Appl. Log.*, 172(3):102903, 2021.

[10] 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`.

[11] Errett Bishop. *Foundations of Constructive Analysis*. McGraw-Hill Book Co., New York, 1967.

[12] 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.

[13] 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.

[14] Douglas Bridges and Fred Richman. *Varieties of Constructive Mathematics*. London Mathematical Society Lecture Note Series. Cambridge University Press, 1987.

[15] Luitzen Egbertus Jan Brouwer. Zur Begründung der intuitionistische Mathematik III. *Mathematische Annalen*, 96:451–488, 1926.

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

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

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

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

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

[21] 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.

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

[23] Thierry Coquand, Henri Lombardi, and Stefan Neuwirth. Constructive theory of ordinals. In M. Benini, O. Beyersdorff, M. Rathjen, and P. Schuster, editors, *Mathematics for Computation (M4C)*. World Scientific, 2022.

[24] Tom de Jong. Agda development on constructive taboos surrounding semidecidability, 2022. Available at cs.bham.ac.uk/~mhe/TypeTopology/SemiDecidable.html.

[25] Tom de Jong and Martín Hötzel Escardó. On small types in univalent foundations. *arXiv:2111.00482v2*, 2022.

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

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

[28] Nachum Dershowitz and Edward M. Reingold. Ordinal arithmetic with list structures. In A. Nerode and M. Taitslin, editors, *Logical Foundations of Computer Science (LFCS 1992)*, volume 620 of *Lecture Notes in Computer Science*, pages 117–138, 1992.

[29] Gabe Dijkstra. *Quotient Inductive-Inductive Definitions.* PhD thesis, University of Nottingham, 2017.

[30] Peter Dybjer and Anton Setzer. A finite axiomatization of inductive-recursive definitions. In *Typed Lambda Calculi and Applications, volume 1581 of Lecture Notes in Computer Science*, pages 129–146. Springer, 1999.

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

[32] Martín Hötzel Escardó. Post under the discussion of "geometric realization" in the IAS Univalent Foundations mailing list, 13 March 2013. Available at https://groups.google.com/g/univalent-foundations/c/SA0dzenV1G4/m/d5iIGdKKNxMJ.

[33] Martín Hötzel Escardó et al. Ordinals in univalent type theory in Agda notation, since 2010. Agda development, HTML rendering available at: https://www.cs.bham.ac.uk/~mhe/TypeTopology/Ordinals.index.html.

[34] Martín Hötzel Escardó and Chuangjie Xu. The inconsistency of a brouwerian continuity principle with the curry-howard interpretation. In Thorsten Altenkirch, editor, *13th International Conference on Typed Lambda Calculi and Applications (TLCA 2015)*, volume 38 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 153–164, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

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

[36] Yannick Forster, Dominik Kirst, and Gert Smolka. On synthetic undecidability in Coq, with an application to the Entscheidungsproblem. In *8th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2019)*, pages 38–51, 2019.

[37] Gerhard Gentzen. Die Widerspruchsfreiheit der reinen Zahlentheorie. *Math. Ann.*, 112:493–565, 1936.

[38] Gerhard Gentzen. Beweisbarkeit und Unbeweisbarkeit von Anfangsfällen der transfiniten Induktion in der reinen Zahlentheorie. *Math. Ann.*, 119:140–161, 1943.

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

[40] Peter Hancock. *Ordinals and Interactive Programs.* PhD thesis, University of Edinburgh, 2000.

[41] Arend Heyting. Infinitistic methods from a finitist point of view. In *Infinitistic Methods. Proceedings of the Symposium on Foundations of Mathematics, Warsaw, 2–9 September 1959*, pages 185–192. Pergamon Press, 1961.

[42] Ambrus Kaposi and András Kovács. A syntax for higher inductive-inductive types. In Hélène Kirchner, editor, *FSCD'18*, volume 108 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:18. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018.

[43] Ambrus Kaposi and András Kovács. Signatures and induction principles for higher inductive-inductive types. *Logical Methods in Compututer Science*, 16(1), 2020.

[44] Ambrus Kaposi, András Kovács, and Thorsten Altenkirch. Constructing quotient inductive-inductive types. In *POPL'19*. ACM, 2019.

[45] Stephen Cole Kleene. On notation for ordinal numbers. *The Journal of Symbolic Logic*, 3(4):150–155, 1938.

[46] Nicolai Kraus, Martín Hötzel Escardó, Thierry Coquand, and Thorsten Altenkirch. Notions of anonymous existence in Martin-Löf type theory. *Logical Methods in Computer Science*, Volume 13, Issue 1, 2017. In the special issue of TLCA'13.

[47] Nicolai Kraus, Fredrik Nordvall Forsberg, and Chuangjie Xu. Connecting constructive notions of ordinals in homotopy type theory. In Filippo Bonchi and Simon J. Puglisi, editors, *46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021, August 23-27, 2021, Tallinn, Estonia*, volume 202 of *LIPIcs*, pages 70:1–70:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

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

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

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

[51] Per Martin-Löf. *Notes on Constructive Mathematics*. Stockholm, Almqvist & Wiksell, 1970.

[52] Ray Mines, Fred Richman, and Wim Ruitenburg. *A course in constructive algebra*. Springer-Verlag, 1988.

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

[54] 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.

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

[56] Kurt Schütte. *Proof Theory*. Springer, 1977.

[57] Helmut Schwichtenberg. Proofs as programs. In P. Aczel, H. Simmons, and S.S. Wainer, editors, *Proof Theory: A Selection of Papers from the Leeds Proof Theory Programme 1990*, pages 81–113. Cambridge University Press, 1992.

[58] Artem Shinkarov. Ordinals in Cantor Normal Form, 2020. Agda development, available at https://github.com/ashinkarov/agda-ordinals.

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

[60] Gaisi Takeuti. *Proof Theory*. North-Holland, 2 edition, 1987.

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

[62] Paul Taylor. *Practical Foundations of Mathematics*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1999.

[63] The agda/cubical development team. The agda/cubical library, 2018–.

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

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

[66] 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.

[67] Stanley S. Wainer. Ordinal recursion, and a refinement of the extended Grzegorczyk hierarchy. *Journal of Symbolic Logic*, 37(2):281–292, 1972.