Ordinal notation systems for ordinals below ε_0 in HoTT

Fredrik Nordvall Forsberg
University of Strathclyde, Glasgow

Joint work with Chuangjie Xu and Nicolai Kraus

Foundations and Applications of Univalent Mathematics
Herrsching, 20 December 2019

Ordinals, classically in set theory

Definition

A set α is an ordinal if it is transitive and $\in$ is well-founded on α:

- $x \in \alpha \to x \subseteq \alpha$,
- Every nonempty $X \subseteq \alpha$ has an $\in$-least element.

(Obviously too strong constructively!)

Ordinals, classically in set theory

Definition

A set α is an ordinal if it is transitive and $\in$ is well-founded on α:

- $x \in \alpha \to x \subseteq \alpha$,
- Every nonempty $X \subseteq \alpha$ has an $\in$-least element.

(Obviously too strong constructively!)

This makes $\in$ a strict total order on α; we often write $<$ for $\in$.

Ordinals, classically in set theory

Definition

A set α is an ordinal if it is transitive and $\in$ is well-founded on α:

- $x \in \alpha \to x \subseteq \alpha$,
- Every nonempty $X \subseteq \alpha$ has an $\in$-least element.

(Obviously too strong constructively!)

This makes $\in$ a strict total order on α; we often write $<$ for $\in$.

Important property: there cannot be an infinitely descending sequence of ordinals

$$\alpha_0 > \alpha_1 > \alpha_2 > \dots$$

Ordinals, classically in set theory

Definition

A set α is an ordinal if it is transitive and $\in$ is well-founded on α:

- $x \in \alpha \to x \subseteq \alpha$,
- Every nonempty $X \subseteq \alpha$ has an $\in$-least element.

(Obviously too strong constructively!)

This makes $\in$ a strict total order on α; we often write $<$ for $\in$.

Important property: there cannot be an infinitely descending sequence of ordinals

$$\alpha_0 > \alpha_1 > \alpha_2 > \ldots$$

E.g. already Turing [1949] used ordinals to prove termination of programs.

Building ordinals

- $0 = \emptyset$ is an ordinal;

Building ordinals

- $0 = \emptyset$ is an ordinal;
- $1 = 0 \cup \{0\}$ is an ordinal;

Building ordinals

- $0 = \emptyset$ is an ordinal;
- $1 = 0 \cup \{0\}$ is an ordinal;
- $2 = 1 \cup \{1\}$ is an ordinal (classically);

Building ordinals

- $0 = \emptyset$ is an ordinal;
- $1 = 0 \cup \{0\}$ is an ordinal;
- $2 = 1 \cup \{1\}$ is an ordinal (classically);
- 3, 4, 5, ... are ordinals.

Building ordinals

- $0 = \emptyset$ is an ordinal;
- $1 = 0 \cup \{0\}$ is an ordinal;
- $2 = 1 \cup \{1\}$ is an ordinal (classically);
- 3, 4, 5, … are ordinals.
- $\omega = \bigcup_{n \in \mathbb{N}} n$ is an ordinal.

Building ordinals

- $0 = \emptyset$ is an ordinal;
- $1 = 0 \cup \{0\}$ is an ordinal;
- $2 = 1 \cup \{1\}$ is an ordinal (classically);
- 3, 4, 5, ... are ordinals.
- $\omega = \bigcup_{n \in \mathbb{N}} n$ is an ordinal.
- $\omega + 1 = \omega \cup \{\omega\}$ is an ordinal;

Building ordinals

- $0 = \emptyset$ is an ordinal;
- $1 = 0 \cup \{0\}$ is an ordinal;
- $2 = 1 \cup \{1\}$ is an ordinal (classically);
- 3, 4, 5, ... are ordinals.
- $\omega = \bigcup_{n \in \mathbb{N}} n$ is an ordinal.
- $\omega + 1 = \omega \cup \{\omega\}$ is an ordinal;
- $\omega + 2$, $\omega + 3$, ... are ordinals;

Building ordinals

- $0 = \emptyset$ is an ordinal;
- $1 = 0 \cup \{0\}$ is an ordinal;
- $2 = 1 \cup \{1\}$ is an ordinal (classically);
- 3, 4, 5, ... are ordinals.
- $\omega = \bigcup_{n \in \mathbb{N}} n$ is an ordinal.
- $\omega + 1 = \omega \cup \{\omega\}$ is an ordinal;
- $\omega + 2$, $\omega + 3$, ... are ordinals;
- $\omega \cdot 2 = \bigcup_{n < \omega} (\omega + n)$ is an ordinal.

Building ordinals $\cup$ { Building ordinals }

- $\omega \cdot 2 = \bigcup_{n<\omega} (\omega + n)$ is an ordinal.

Building ordinals $\cup$ { Building ordinals }

- $\omega \cdot 2 = \bigcup_{n<\omega} (\omega + n)$ is an ordinal.
- $\omega \cdot 2$, $\omega \cdot 3$, $\ldots$ are ordinals;

Building ordinals $\cup$ { Building ordinals }

- $\omega \cdot 2 = \bigcup_{n<\omega} (\omega + n)$ is an ordinal.
- $\omega \cdot 2$, $\omega \cdot 3$, $\ldots$ are ordinals;
- $\omega^2 = \omega \cdot \omega = \bigcup_{n<\omega} (\omega \cdot n)$ is an ordinal.

Building ordinals $\cup$ { Building ordinals }

- $\omega \cdot 2 = \bigcup_{n<\omega} (\omega + n)$ is an ordinal.
- $\omega \cdot 2$, $\omega \cdot 3$, $\ldots$ are ordinals;
- $\omega^2 = \omega \cdot \omega = \bigcup_{n<\omega} (\omega \cdot n)$ is an ordinal.
- $\omega^2 \cdot 2$, $\omega^2 \cdot 3$, $\ldots$ are ordinals;

Building ordinals $\cup$ { Building ordinals }

- $\omega \cdot 2 = \bigcup_{n<\omega} (\omega + n)$ is an ordinal.
- $\omega \cdot 2$, $\omega \cdot 3$, ... are ordinals;
- $\omega^2 = \omega \cdot \omega = \bigcup_{n<\omega} (\omega \cdot n)$ is an ordinal.
- $\omega^2 \cdot 2$, $\omega^2 \cdot 3$, ... are ordinals;
- $\omega^3 = \bigcup_{n<\omega} (\omega^2 \cdot n)$ is an ordinal.

Building ordinals $\cup$ $\{$ Building ordinals $\}$

- $\omega \cdot 2 = \bigcup_{n<\omega} (\omega + n)$ is an ordinal.
- $\omega \cdot 2$, $\omega \cdot 3$, $\ldots$ are ordinals;
- $\omega^2 = \omega \cdot \omega = \bigcup_{n<\omega} (\omega \cdot n)$ is an ordinal.
- $\omega^2 \cdot 2$, $\omega^2 \cdot 3$, $\ldots$ are ordinals;
- $\omega^3 = \bigcup_{n<\omega} (\omega^2 \cdot n)$ is an ordinal.
- ω^4, ω^5, $\ldots$ are ordinals;

Building ordinals $\cup$ { Building ordinals }

- $\omega \cdot 2 = \bigcup_{n<\omega} (\omega + n)$ is an ordinal.
- $\omega \cdot 2$, $\omega \cdot 3$, ... are ordinals;
- $\omega^2 = \omega \cdot \omega = \bigcup_{n<\omega} (\omega \cdot n)$ is an ordinal.
- $\omega^2 \cdot 2$, $\omega^2 \cdot 3$, ... are ordinals;
- $\omega^3 = \bigcup_{n<\omega} (\omega^2 \cdot n)$ is an ordinal.
- ω^4, ω^5, ... are ordinals;
- $\omega^\omega = \bigcup_{n<\omega} \omega^n$ is an ordinal.

Building ordinals $\cup$ { Building ordinals }

- $\omega \cdot 2 = \bigcup_{n<\omega} (\omega + n)$ is an ordinal.
- $\omega \cdot 2$, $\omega \cdot 3$, ... are ordinals;
- $\omega^2 = \omega \cdot \omega = \bigcup_{n<\omega} (\omega \cdot n)$ is an ordinal.
- $\omega^2 \cdot 2$, $\omega^2 \cdot 3$, ... are ordinals;
- $\omega^3 = \bigcup_{n<\omega} (\omega^2 \cdot n)$ is an ordinal.
- ω^4, ω^5, ... are ordinals;
- $\omega^\omega = \bigcup_{n<\omega} \omega^n$ is an ordinal.
- ω^ω, ω^{ω^ω}, ... are ordinals;

Building ordinals $\cup$ { Building ordinals }

- $\omega \cdot 2 = \bigcup_{n<\omega} (\omega + n)$ is an ordinal.
- $\omega \cdot 2$, $\omega \cdot 3$, ... are ordinals;
- $\omega^2 = \omega \cdot \omega = \bigcup_{n<\omega} (\omega \cdot n)$ is an ordinal.
- $\omega^2 \cdot 2$, $\omega^2 \cdot 3$, ... are ordinals;
- $\omega^3 = \bigcup_{n<\omega} (\omega^2 \cdot n)$ is an ordinal.
- ω^4, ω^5, ... are ordinals;
- $\omega^\omega = \bigcup_{n<\omega} \omega^n$ is an ordinal.
- ω^ω, ω^{ω^ω}, ... are ordinals;
- $\bigcup\{\omega^\omega, \omega^{\omega^\omega}, \omega^{\omega^{\omega^\omega}}, \ldots\}$ is an ordinal.

Building ordinals $\cup$ { Building ordinals }

- $\omega \cdot 2 = \bigcup_{n<\omega} (\omega + n)$ is an ordinal.
- $\omega \cdot 2$, $\omega \cdot 3$, ... are ordinals;
- $\omega^2 = \omega \cdot \omega = \bigcup_{n<\omega} (\omega \cdot n)$ is an ordinal.
- $\omega^2 \cdot 2$, $\omega^2 \cdot 3$, ... are ordinals;
- $\omega^3 = \bigcup_{n<\omega} (\omega^2 \cdot n)$ is an ordinal.
- ω^4, ω^5, ... are ordinals;
- $\omega^\omega = \bigcup_{n<\omega} \omega^n$ is an ordinal.
- ω^ω, ω^{ω^ω}, ... are ordinals;
- $\varepsilon_0 = \bigcup\{\omega^\omega, \omega^{\omega^\omega}, \omega^{\omega^{\omega^\omega}}, \ldots\}$ is an ordinal.

$\mathcal{E}_0$

ε_0

ε_0 is the least solution to the equation $\alpha = \omega^\alpha$.

ε_0

ε_0 is the least solution to the equation $\alpha = \omega^\alpha$.

Fact (Cantor Normal Form)

Every ordinal α can be written uniquely as

$$\alpha = \omega^{\beta_1} + \omega^{\beta_2} + \cdots + \omega^{\beta_n}$$

for some $\beta_1 \geq \beta_2 \geq \cdots \geq \beta_n$.

ε_0

ε_0 is the least solution to the equation $\alpha = \omega^\alpha$.

Fact (Cantor Normal Form)

Every ordinal α can be written uniquely as

$$\alpha = \omega^{\beta_1} + \omega^{\beta_2} + \cdots + \omega^{\beta_n}$$

for some $\beta_1 \geq \beta_2 \geq \cdots \geq \beta_n$.

In particular, $\varepsilon_0 = \omega^{\varepsilon_0}$, so we can take $\beta_1 = \varepsilon_0$.

ε_0

ε_0 is the least solution to the equation $\alpha = \omega^\alpha$.

Fact (Cantor Normal Form)

Every ordinal α can be written uniquely as

$$\alpha = \omega^{\beta_1} + \omega^{\beta_2} + \cdots + \omega^{\beta_n}$$

for some $\beta_1 \geq \beta_2 \geq \cdots \geq \beta_n$.

In particular, $\varepsilon_0 = \omega^{\varepsilon_0}$, so we can take $\beta_1 = \varepsilon_0$.

But, for $\alpha < \varepsilon_0$, we have $\beta_i < \alpha$ for every i.

ε_0

ε_0 is the least solution to the equation $\alpha = \omega^\alpha$.

Fact (Cantor Normal Form)

Every ordinal α can be written uniquely as

$$\alpha = \omega^{\beta_1} + \omega^{\beta_2} + \cdots + \omega^{\beta_n}$$

for some $\beta_1 \geq \beta_2 \geq \cdots \geq \beta_n$.

In particular, $\varepsilon_0 = \omega^{\varepsilon_0}$, so we can take $\beta_1 = \varepsilon_0$.

But, for $\alpha < \varepsilon_0$, we have $\beta_i < \alpha$ for every i.

Hence if we compute the Cantor Normal Form

$$\beta_i = \omega^{\gamma_1} + \omega^{\gamma_2} + \cdots + \omega^{\gamma_m}$$

and so on, we get decreasing sequences

$$\alpha > \beta_i > \gamma_j > \ldots$$

which must terminate.

ε_0

ε_0 is the least solution to the equation $\alpha = \omega^\alpha$.

Fact (Cantor Normal Form)

Every ordinal α can be written uniquely as

$$\alpha = \omega^{\beta_1} + \omega^{\beta_2} + \cdots + \omega^{\beta_n}$$

for some $\beta_1 \geq \beta_2 \geq \cdots \geq \beta_n$.

In particular, $\varepsilon_0 = \omega^{\varepsilon_0}$, so we can take $\beta_1 = \varepsilon_0$.

But, for $\alpha < \varepsilon_0$, we have $\beta_i < \alpha$ for every i.

Hence if we compute the Cantor Normal Form

$$\beta_i = \omega^{\gamma_1} + \omega^{\gamma_2} + \cdots + \omega^{\gamma_m}$$

and so on, we get decreasing sequences

$$\alpha > \beta_i > \gamma_j > \ldots$$

which must terminate. This gives a finite representation of α!

Ordinal notation systems for ordinals below ε_0

Cantor Normal Form gives a finite and simple notation for ordinals α below ε_0:

Ordinal notation systems for ordinals below ε_0

Cantor Normal Form gives a finite and simple notation for ordinals α below ε_0:

- α is either 0, or

Ordinal notation systems for ordinals below ε_0

Cantor Normal Form gives a finite and simple notation for ordinals α below ε_0:

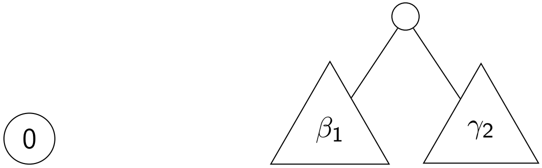
- α is either 0, or
- represented by two ordinals $\alpha = \omega^{\beta_1} + \gamma_2$.

Ordinal notation systems for ordinals below ε_0

Cantor Normal Form gives a finite and simple notation for ordinals α below ε_0:

- α is either 0, or
- represented by two ordinals $\alpha = \omega^{\beta_1} + \gamma_2$.

Simply binary trees!

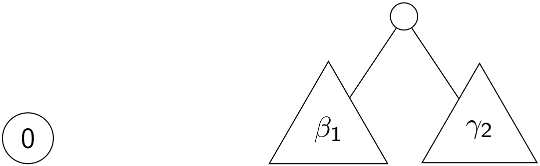


Ordinal notation systems for ordinals below ε_0

Cantor Normal Form gives a finite and simple notation for ordinals α below ε_0:

- α is either 0, or
- represented by two ordinals $\alpha = \omega^{\beta_1} + \gamma_2$.

Simply binary trees!



But: uniqueness of representation has been lost. How can we recover this?


A mutual approach

An inductive-inductive-recursive definition

We simultaneously define

$$\mathsf{data}\ \mathsf{MutualOrd} : \mathsf{Type}_0$$
$$\mathsf{data}\ \_<\_ : \mathsf{MutualOrd} \to \mathsf{MutualOrd} \to \mathsf{Type}_0$$
$$\mathsf{fst} : \mathsf{MutualOrd} \to \mathsf{MutualOrd}$$

MutualOrd

$$
\begin{array}{l}
\mathsf{data}\ \mathsf{MutualOrd}\ \mathsf{where} \\
\quad \mathbf{0} : \mathsf{MutualOrd} \\
\quad \omega\text{^}\_+\_[\_] : (a\ b : \mathsf{MutualOrd}) \to a \geq \mathsf{fst}\ b \to \mathsf{MutualOrd}
\end{array}
$$

MutualOrd

```
data MutualOrd where
  0 : MutualOrd
  ω^_+_[_] : (a b : MutualOrd) → a ≥ fst b → MutualOrd
```

where $a \geq b = a > b \uplus a \equiv b$.

MutualOrd

```
data MutualOrd where
  0 : MutualOrd
  ω^_+_[_] : (a b : MutualOrd) → a ≥ fst b → MutualOrd
```

where $a \geq b = a > b \uplus a \equiv b$.

```
data _<_ where
  <₁ : 0 < ω^ a + b [ r ]
  <₂ : a < c → ω^ a + b [ r ] < ω^ c + d [ s ]
  <₃ : a ≡ c → b < d → ω^ a + b [ r ] < ω^ c + d [ s ]
```

MutualOrd

$$
\begin{array}{l}
\textsf{data MutualOrd where} \\
\quad \mathbf{0} : \textsf{MutualOrd} \\
\quad \omega\hat{}\ \_+\_[\_] : (a\ b : \textsf{MutualOrd}) \to a \geq \textsf{fst}\ b \to \textsf{MutualOrd}
\end{array}
$$

where $a \geq b = a > b \uplus a \equiv b$.

$$
\begin{array}{l}
\textsf{data}\ \_<\_\ \textsf{where} \\
\quad <_1 : \mathbf{0} < \omega\hat{}\ a + b\,[\,r\,] \\
\quad <_2 : a < c \to \omega\hat{}\ a + b\,[\,r\,] < \omega\hat{}\ c + d\,[\,s\,] \\
\quad <_3 : a \equiv c \to b < d \to \omega\hat{}\ a + b\,[\,r\,] < \omega\hat{}\ c + d\,[\,s\,]
\end{array}
$$

$$
\begin{array}{l}
\textsf{fst}\ \mathbf{0} = \mathbf{0} \\
\textsf{fst}\ (\omega\hat{}\ a + \_\ [\ \_\ ]) = a
\end{array}
$$

MutualOrd

$$\begin{array}{l}
\textbf{data}\ \mathsf{MutualOrd}\ \textbf{where} \\
\quad \mathbf{0} : \mathsf{MutualOrd} \\
\quad \omega\verb|^|_+_[_] : (a\ b : \mathsf{MutualOrd}) \to a \geq \mathsf{fst}\ b \to \mathsf{MutualOrd}
\end{array}$$

where $a \geq b = a > b \uplus a \equiv b$.

$$\begin{array}{l}
\textbf{data}\ _<_\ \textbf{where} \\
\quad <_1 : \mathbf{0} < \omega\verb|^|\ a + b\ [\ r\ ] \\
\quad <_2 : a < c \to \omega\verb|^|\ a + b\ [\ r\ ] < \omega\verb|^|\ c + d\ [\ s\ ] \\
\quad <_3 : a \equiv c \to b < d \to \omega\verb|^|\ a + b\ [\ r\ ] < \omega\verb|^|\ c + d\ [\ s\ ]
\end{array}$$

$$\begin{array}{l}
\mathsf{fst}\ \mathbf{0} = \mathbf{0} \\
\mathsf{fst}\ (\omega\verb|^|\ a + _\ [\ _\ ]) = a
\end{array}$$

Remark: there is an equivalent non-inductive-recursive definition where we define the graph of fst inductively.

Examples

- $\mathbf{1} = \omega\hat{}\ \mathbf{0} + \mathbf{0}\ [\ \mathsf{inj}_2\ \mathsf{refl}\ ]$
- $\omega = \omega\hat{}\ \mathbf{1} + \mathbf{0}\ [\ \mathsf{inj}_1\ <_1\ ]$
- $\omega\hat{}\langle\ a\ \rangle = \omega\hat{}\ a + \mathbf{0}\ [\ \geq\mathbf{0}\ ]$

Basic properties

Proposition

$\_<\_$ *is proof-irrelevant.*

Basic properties

Proposition

$\_<\_$ is proof-irrelevant.

Proof.

We simultaneously define

$$
\begin{aligned}
&\mathsf{MutualOrdIsSet} : \mathsf{isSet}\ \mathsf{MutualOrd} \\
&\mathsf{<IsPropValued} : \mathsf{isProp}\ (a < b) \\
&\mathsf{MutualOrd}^{=} : \{r : a \geq \mathsf{fst}\ b\}\ \{s : c \geq \mathsf{fst}\ d\} \to a \equiv c \to b \equiv d \\
&\qquad\qquad \to \omega\hat{}\ a + b\ [\ r\ ] \equiv \omega\hat{}\ c + d\ [\ s\ ]
\end{aligned}
$$

$\square$

Basic properties

Proposition

$\_<\_$ is proof-irrelevant.

Proof.

We simultaneously define

$$\begin{aligned}
&\mathsf{MutualOrdIsSet} : \mathsf{isSet}\ \mathsf{MutualOrd} \\
&\mathsf{<IsPropValued} : \mathsf{isProp}\ (a < b) \\
&\mathsf{MutualOrd}^{=} : \{r : a \geq \mathsf{fst}\ b\}\ \{s : c \geq \mathsf{fst}\ d\} \to a \equiv c \to b \equiv d \\
&\qquad\qquad \to \omega\hat{}\ a + b\ [\ r\ ] \equiv \omega\hat{}\ c + d\ [\ s\ ]
\end{aligned}$$

□

Proposition

$\_<\_$ is trichotomous.

Proof.

We define

$$\mathsf{<\text{-}tri} : (a\ b : \mathsf{MutualOrd}) \to a < b \uplus a \geq b$$

using case distinctions on all subterms. □

Ordinal addition

Addition on ordinals is famously non-commutative

Ordinal addition

Addition on ordinals is famously non-commutative:

$$1 + \omega = \omega$$

Ordinal addition

Addition on ordinals is famously non-commutative:

$$1 + \omega = \omega < \omega + 1$$

Ordinal addition

Addition on ordinals is famously non-commutative:

$$1 + \omega = \omega < \omega + 1$$

In general, if $\gamma < \omega^\beta$ then $\gamma + \omega^\beta = \omega^\beta$.

Ordinal addition

Addition on ordinals is famously non-commutative:

$$1 + \omega = \omega < \omega + 1$$

In general, if $\gamma < \omega^\beta$ then $\gamma + \omega^\beta = \omega^\beta$.

In particular, if $\alpha < \beta$ then $\omega^\alpha < \omega^\beta$, hence $\omega^\alpha + \omega^\beta = \omega^\beta$.

Ordinal addition

Addition on ordinals is famously non-commutative:

$$1 + \omega = \omega < \omega + 1$$

In general, if $\gamma < \omega^\beta$ then $\gamma + \omega^\beta = \omega^\beta$.

In particular, if $\alpha < \beta$ then $\omega^\alpha < \omega^\beta$, hence $\omega^\alpha + \omega^\beta = \omega^\beta$.

We now want to implement addition on MutualOrd. We simultaneously define

$$\begin{aligned} \_+\_ &: \mathsf{MutualOrd} \to \mathsf{MutualOrd} \to \mathsf{MutualOrd} \\ \geq\mathsf{fst}+ &: \{a : \mathsf{MutualOrd}\}\ (b\ c : \mathsf{MutualOrd}) \\ &\to a \geq \mathsf{fst}\ b \to a \geq \mathsf{fst}\ c \to a \geq \mathsf{fst}\ (b + c) \end{aligned}$$

Addition on MutualOrd

Remember: if $\alpha < \beta$ then $\omega^\alpha + \omega^\beta = \omega^\beta$.

Addition on MutualOrd

Remember: if $\alpha < \beta$ then $\omega^\alpha + \omega^\beta = \omega^\beta$.

```
0 + b = b
a + 0 = a
(ω^ a + c [ r ]) + (ω^ b + d [ s ]) with <-tri a b
... | inj₁ a<b = ω^ b + d [ s ]
... | inj₂ a≥b = ω^ a + (c + ω^ b + d [ s ]) [ ≥fst+ c _ r a≥b ]
```

Addition on MutualOrd

Remember: if $\alpha < \beta$ then $\omega^\alpha + \omega^\beta = \omega^\beta$.

```
0 + b = b
a + 0 = a
(ω^ a + c [ r ]) + (ω^ b + d [ s ]) with <-tri a b
... | inj₁ a<b = ω^ b + d [ s ]
... | inj₂ a≥b = ω^ a + (c + ω^ b + d [ s ]) [ ≥fst+ c _ r a≥b ]

≥fst+ 0 _ r s = s
≥fst+ (ω^ _ + _ [ _ ]) 0 r s = r
≥fst+ (ω^ b + _ [ _ ]) (ω^ c + _ [ _ ]) r s with <-tri b c
... | inj₁ b<c = s
... | inj₂ b≥c = r
```

Multiplication on MutualOrd

$$
\begin{aligned}
&\_\cdot\_ : \mathrm{MutualOrd} \to \mathrm{MutualOrd} \to \mathrm{MutualOrd} \\
&\mathbf{0} \cdot b = \mathbf{0} \\
&a \cdot \mathbf{0} = \mathbf{0} \\
&a \cdot (\omega\hat{}\ \mathbf{0} + d\,[\,r\,]) = a + a \cdot d \\
&(\omega\hat{}\ a + c\,[\,r\,]) \cdot (\omega\hat{}\ b + d\,[\,s\,]) = \\
&\quad \mathrm{M}.\omega\hat{}\langle\, a + b\,\rangle + (\omega\hat{}\ a + c\,[\,r\,] \cdot d)
\end{aligned}
$$

Multiplication on MutualOrd

$$
\begin{aligned}
&\_\cdot\_ : \text{MutualOrd} \to \text{MutualOrd} \to \text{MutualOrd} \\
&\mathbf{0} \cdot b = \mathbf{0} \\
&a \cdot \mathbf{0} = \mathbf{0} \\
&a \cdot (\omega\hat{}\ \mathbf{0} + d\ [\ r\ ]) = a + a \cdot d \\
&(\omega\hat{}\ a + c\ [\ r\ ]) \cdot (\omega\hat{}\ b + d\ [\ s\ ]) = \\
&\quad \text{M.}\omega\hat{}\langle\ a + b\ \rangle + (\omega\hat{}\ a + c\ [\ r\ ] \cdot d)
\end{aligned}
$$

Note: All in terms of previous operations, so no simultaneous lemma needed.


A higher inductive approach

Uniqueness by making things the same

We want to avoid redundant representations of ordinals

$$\alpha = \omega^{\beta_1} + \omega^{\beta_2} + \cdots + \omega^{\beta_n}$$

Uniqueness by making things the same

We want to avoid redundant representations of ordinals

$$\alpha = \omega^{\beta_1} + \omega^{\beta_2} + \cdots + \omega^{\beta_n}$$

With a mutual approach, we could require $\beta_1 \geq \beta_2 \geq \ldots \geq \beta_n$, hence ensuring uniqueness of the list $[\beta_1, \ldots, \beta_n]$.

Uniqueness by making things the same

We want to avoid redundant representations of ordinals

$$\alpha = \omega^{\beta_1} + \omega^{\beta_2} + \cdots + \omega^{\beta_n}$$

With a mutual approach, we could require $\beta_1 \geq \beta_2 \geq \ldots \geq \beta_n$, hence ensuring uniqueness of the list $[\beta_1, \ldots, \beta_n]$.

Another option: quotient out the difference by identifying different permutations of the exponents

$$\omega^{\beta_1} + \omega^{\beta_2} \equiv \omega^{\beta_2} + \omega^{\beta_1}$$

Uniqueness by making things the same

We want to avoid redundant representations of ordinals

$$\alpha = \omega^{\beta_1} + \omega^{\beta_2} + \cdots + \omega^{\beta_n}$$

With a mutual approach, we could require $\beta_1 \geq \beta_2 \geq \ldots \geq \beta_n$, hence ensuring uniqueness of the list $[\beta_1, \ldots, \beta_n]$.

Another option: quotient out the difference by identifying different permutations of the exponents

$$\omega^{\beta_1} + \omega^{\beta_2} \equiv \omega^{\beta_2} + \omega^{\beta_1}$$

That is, we use a quotient inductive type.

A higher inductive appoach

```
data HITOrd : Type₀ where
  0 : HITOrd
  ω^_⊕_ : HITOrd → HITOrd → HITOrd
  swap : ∀ a b c → ω^ a ⊕ ω^ b ⊕ c ≡ ω^ b ⊕ ω^ a ⊕ c
  trunc : isSet HITOrd
```

(cf. finite multisets as a HIT [Licata, 2014]).

Example

```
example : (a b c : HITOrd)
  → ω^ a ⊕ ω^ b ⊕ ω^ c ⊕ 0 ≡ ω^ c ⊕ ω^ b ⊕ ω^ a ⊕ 0
example a b c = begin
  ω^ a ⊕ ω^ b ⊕ ω^ c ⊕ 0 ≡⟨ swap a b _ ⟩
  ω^ b ⊕ ω^ a ⊕ ω^ c ⊕ 0 ≡⟨ cong (ω^ b ⊕_) (swap a c _) ⟩
  ω^ b ⊕ ω^ c ⊕ ω^ a ⊕ 0 ≡⟨ swap b c _ ⟩
  ω^ c ⊕ ω^ b ⊕ ω^ a ⊕ 0 □
```

Arithmetic on HITOrd

Because every function out of HITOrd must respect swap, it is convenient to define **commutative** operations on HITOrd.

Arithmetic on HITOrd

Because every function out of HITOrd must respect swap, it is convenient to define **commutative** operations on HITOrd.

For arithmetic, these are the so-called Hessenberg sum and product [Hessenberg, 1906].

Hessenberg sum

```
_⊕_ : HITOrd → HITOrd → HITOrd
0                ⊕ y = y
(ω^ a ⊕ b)       ⊕ y = ω^ a ⊕ (b ⊕ y)
(swap a b c i)   ⊕ y = swap a b (c ⊕ y) i
(trunc p q i j)  ⊕ y = trunc (cong (_⊕ y) p) (cong (_⊕ y) q) i j
```

Hessenberg sum

$$
\begin{array}{l}
\_\oplus\_ : \mathsf{HITOrd} \to \mathsf{HITOrd} \to \mathsf{HITOrd} \\
\mathbf{0} \qquad\qquad\qquad\quad \oplus\ y = y \\
(\omega\hat{}\ a \oplus b) \quad\ \ \oplus\ y = \omega\hat{}\ a \oplus (b \oplus y) \\
(\mathsf{swap}\ a\ b\ c\ i) \oplus y = \mathsf{swap}\ a\ b\ (c \oplus y)\ i \\
(\mathsf{trunc}\ p\ q\ i\ j) \oplus y = \mathsf{trunc}\ (\mathsf{cong}\ (\_\oplus y)\ p)\ (\mathsf{cong}\ (\_\oplus y)\ q)\ i\ j
\end{array}
$$

In the swap case, we have to construct a path

$$\omega\hat{}\ a \oplus \omega\hat{}\ b \oplus (c \oplus y) \equiv \omega\hat{}\ b \oplus \omega\hat{}\ a \oplus (c \oplus y)$$

Hessenberg sum

$$
\begin{array}{l}
\_\oplus\_ : \mathsf{HITOrd} \to \mathsf{HITOrd} \to \mathsf{HITOrd} \\
\mathbf{0} \qquad\qquad\qquad\quad \oplus\ y = y \\
(\omega\hat{}\ a \oplus b) \quad\ \ \oplus\ y = \omega\hat{}\ a \oplus (b \oplus y) \\
(\mathsf{swap}\ a\ b\ c\ i) \oplus y = \mathsf{swap}\ a\ b\ (c \oplus y)\ i \\
(\mathsf{trunc}\ p\ q\ i\ j) \oplus y = \mathsf{trunc}\ (\mathsf{cong}\ (\_\oplus y)\ p)\ (\mathsf{cong}\ (\_\oplus y)\ q)\ i\ j
\end{array}
$$

In the swap case, we have to construct a path

$$\omega\hat{}\ a \oplus \omega\hat{}\ b \oplus (c \oplus y) \equiv \omega\hat{}\ b \oplus \omega\hat{}\ a \oplus (c \oplus y)$$

Proposition

$\_\oplus\_$ *is commutative.*

Which approach is better?

Which approach is better?

Both!

Which approach is better?

Both!

Depending on the application, e.g. the mutual approach for properties of the order, the HIT approach for commutative operations.

Which approach is better?

Both!

Depending on the application, e.g. the mutual approach for properties of the order, the HIT approach for commutative operations.

Even better:

Theorem

$\mathit{MutualOrd}$ and HITOrd are equivalent, i.e. there is a proof $\mathit{M{\simeq}H} : \mathit{MutualOrd} \simeq \mathit{HITOrd}$.

Which approach is better?

Both!

Depending on the application, e.g. the mutual approach for properties of the order, the HIT approach for commutative operations.

Even better:

Theorem

MutualOrd and HITOrd are equivalent, i.e. there is a proof $M{\simeq}H$: $MutualOrd \simeq HITOrd$.

Corollary

MutualOrd and HITOrd are identical, i.e. there is a proof $M{\equiv}H$: $MutualOrd \equiv HITOrd$.

MutualOrd and HITOrd are equivalent

$$\begin{tikzcd}
\mathsf{MutualOrd} \arrow[r, bend right] & \mathsf{HITOrd} \arrow[l, bend right]
\end{tikzcd}$$

MutualOrd and HITOrd are equivalent

$$\begin{tikzcd} \mathsf{MutualOrd} \arrow[r, bend right, "\text{``forget order''}"'] & \mathsf{HITOrd} \arrow[l, bend right] \end{tikzcd}$$

MutualOrd and HITOrd are equivalent

$$
\begin{tikzcd}
\mathsf{MutualOrd} \arrow[r, bend right, "\text{``forget order''}"'] & \mathsf{HITOrd} \arrow[l, bend right, "\text{``insert''}"']
\end{tikzcd}
$$

Operations via univalence

By using univalence, we can transport operations and proofs between MutualOrd and HITOrd.

Operations via univalence

By using univalence, we can transport operations and proofs between MutualOrd and HITOrd.

$$
\begin{aligned}
&\_<^{\mathsf{H}}\_ : \mathsf{HITOrd} \to \mathsf{HITOrd} \to \mathsf{Type}_0 \\
&\_<^{\mathsf{H}}\_ = \mathsf{transport}\ (\lambda\ i \to \mathsf{M}{\equiv}\mathsf{H}\ i \to \mathsf{M}{\equiv}\mathsf{H}\ i \to \mathsf{Type}_0)\ \_<\_
\end{aligned}
$$

Operations via univalence

By using univalence, we can transport operations and proofs between MutualOrd and HITOrd.

$$\_<^{\mathsf{H}}\_ : \mathsf{HITOrd} \to \mathsf{HITOrd} \to \mathsf{Type}_0$$
$$\_<^{\mathsf{H}}\_ = \mathsf{transport}\ (\lambda\ i \to \mathsf{M{\equiv}H}\ i \to \mathsf{M{\equiv}H}\ i \to \mathsf{Type}_0)\ \_<\_$$

$$\_\oplus^{\mathsf{M}}\_ : \mathsf{MutualOrd} \to \mathsf{MutualOrd} \to \mathsf{MutualOrd}$$
$$\_\oplus^{\mathsf{M}}\_ = \mathsf{transport}\ (\lambda\ i \to \mathsf{H{\equiv}M}\ i \to \mathsf{H{\equiv}M}\ i \to \mathsf{H{\equiv}M}\ i)\ \_\oplus\_$$

Transporting proofs

We can also transport properties. For instance: define

$$\mathsf{Dec} : (A : \mathsf{Type}\ \ell) \to (A \to A \to \mathsf{Type}\ \ell') \to \mathsf{Type}\ (\ell \sqcup \ell')$$
$$\mathsf{Dec}\ A\ \_<\_ = (x\ y : A) \to x < y \uplus \neg\, x < y$$

Transporting proofs

We can also transport properties. For instance: define

$$\mathsf{Dec} : (A : \mathsf{Type}\ \ell) \to (A \to A \to \mathsf{Type}\ \ell') \to \mathsf{Type}\ (\ell \sqcup \ell')$$
$$\mathsf{Dec}\ A\ \_<\_ = (x\ y : A) \to x < y \uplus \neg\, x < y$$

We can easily prove

$$\mathsf{<\text{-}dec} : \mathsf{Dec\ MutualOrd}\ \_<\_$$

Transporting proofs

We can also transport properties. For instance: define

$$\mathsf{Dec} : (A : \mathsf{Type}\ \ell) \to (A \to A \to \mathsf{Type}\ \ell') \to \mathsf{Type}\ (\ell \sqcup \ell')$$
$$\mathsf{Dec}\ A\ \_{<}\_ = (x\ y : A) \to x < y \uplus \neg\, x < y$$

We can easily prove

$$\mathsf{<\text{-}dec} : \mathsf{Dec}\ \mathsf{MutualOrd}\ \_{<}\_$$

Hence we can construct

$$\mathsf{<^{H}\text{-}dec} : \mathsf{Dec}\ \mathsf{HITOrd}\ \_{<^{H}}\_$$
$$\mathsf{<^{H}\text{-}dec} = \mathsf{transport}\ (\lambda\ i \to \mathsf{Dec}\ (\mathsf{M{\equiv}H}\ i)\ (\mathsf{<Path}\ i))\ \mathsf{<\text{-}dec}$$

where

$$\mathsf{<Path} : \mathsf{PathP}\ (\lambda\ i \to \mathsf{M{\equiv}H}\ i \to \mathsf{M{\equiv}H}\ i \to \mathsf{Type}_0)\ \_{<}\_\ \_{<^{H}}\_$$

is a dependent path from $\_{<}\_$ to $\_{<^{H}}\_$.

It computes!

Define

$$\mathsf{lt} : \mathsf{HITOrd} \to \mathsf{HITOrd} \to \mathsf{Bool}$$
$$\mathsf{lt}\ a\ b = \mathsf{isLeft}\ (<^{\mathsf{H}}\text{-}\mathsf{dec}\ a\ b)$$

for convenience.

It computes!

Define

$$\text{lt} : \text{HITOrd} \to \text{HITOrd} \to \text{Bool}$$
$$\text{lt}\ a\ b = \text{isLeft}\ ({<^{\text{H}}}\text{-dec}\ a\ b)$$

for convenience.

$$\begin{aligned}
&\text{Ex}[{<^{\text{H}}}\text{-decComp}] : \\
&\qquad \text{lt}\ \mathbf{0}\ \mathbf{0} \equiv \text{false} \\
&\quad \times\ \text{lt}\ \text{H}.\omega\ ((\text{H}.\mathbf{1} \oplus \text{H}.\mathbf{1}) \otimes \text{H}.\omega) \equiv \text{true} \\
&\quad \times\ \text{lt}\ (\text{H}.\omega\text{\textasciicircum}\langle\ \text{H}.\omega\ \rangle)\ (\text{H}.\omega\text{\textasciicircum}\langle\ \text{H}.\mathbf{1} +^{\text{H}} \text{H}.\omega\ \rangle) \equiv \text{false} \\
&\quad \times\ \text{lt}\ (\text{H}.\omega\text{\textasciicircum}\langle\ \text{H}.\omega\ \rangle)\ (\text{H}.\omega\text{\textasciicircum}\langle\ \text{H}.\mathbf{1} \oplus \text{H}.\omega\ \rangle) \equiv \text{true} \\
&\text{Ex}[{<^{\text{H}}}\text{-decComp}] = (\text{refl}\ ,\ \text{refl}\ ,\ \text{refl}\ ,\ \text{refl})
\end{aligned}$$

It computes!

Define

$$\mathsf{lt} : \mathsf{HITOrd} \to \mathsf{HITOrd} \to \mathsf{Bool}$$
$$\mathsf{lt}\ a\ b = \mathsf{isLeft}\ ({<^{\mathsf{H}}}\text{-}\mathsf{dec}\ a\ b)$$

for convenience.

$$\begin{aligned}
&\mathsf{Ex}[{<^{\mathsf{H}}}\text{-}\mathsf{decComp}] : \\
&\qquad \mathsf{lt}\ \mathbf{0}\ \mathbf{0} \equiv \mathsf{false} \\
&\quad \times\ \mathsf{lt}\ \mathsf{H}.\omega\ ((\mathsf{H}.\mathbf{1} \oplus \mathsf{H}.\mathbf{1}) \otimes \mathsf{H}.\omega) \equiv \mathsf{true} \\
&\quad \times\ \mathsf{lt}\ (\mathsf{H}.\omega\text{\textasciicircum}\langle\ \mathsf{H}.\omega\ \rangle)\ (\mathsf{H}.\omega\text{\textasciicircum}\langle\ \mathsf{H}.\mathbf{1} +^{\mathsf{H}} \mathsf{H}.\omega\ \rangle) \equiv \mathsf{false} \\
&\quad \times\ \mathsf{lt}\ (\mathsf{H}.\omega\text{\textasciicircum}\langle\ \mathsf{H}.\omega\ \rangle)\ (\mathsf{H}.\omega\text{\textasciicircum}\langle\ \mathsf{H}.\mathbf{1} \oplus \mathsf{H}.\omega\ \rangle) \equiv \mathsf{true} \\
&\mathsf{Ex}[{<^{\mathsf{H}}}\text{-}\mathsf{decComp}] = (\mathsf{refl}\ ,\ \mathsf{refl}\ ,\ \mathsf{refl}\ ,\ \mathsf{refl})
\end{aligned}$$

$$\mathsf{Ex}[\oplus^{\mathsf{M}}\mathsf{Comp}] : \mathsf{M}.\mathbf{1} \oplus^{\mathsf{M}} \mathsf{M}.\omega \equiv \mathsf{M}.\omega + \mathsf{M}.\mathbf{1}$$
$$\mathsf{Ex}[\oplus^{\mathsf{M}}\mathsf{Comp}] = \mathsf{refl}$$

Summary and outlook

Conclusions

- Summary: Using mutual definitions and higher inductive types to faithfully represent ordinals in HoTT.

Conclusions

- Summary: Using mutual definitions and higher inductive types to faithfully represent ordinals in HoTT.

- Moral: Define operations on the data structure that is suited for the operation (then transport across with univalence).

Conclusions

- Summary: Using mutual definitions and higher inductive types to faithfully represent ordinals in HoTT.
- Moral: Define operations on the data structure that is suited for the operation (then transport across with univalence).
- Also: Can prove e.g. transfinite induction.

Conclusions

- Summary: Using mutual definitions and higher inductive types to faithfully represent ordinals in HoTT.
- Moral: Define operations on the data structure that is suited for the operation (then transport across with univalence).
- Also: Can prove e.g. transfinite induction.
- Future work: Going beyond ε_0 using a higher inductive type of Brouwer ordinals.

Conclusions

- Summary: Using mutual definitions and higher inductive types to faithfully represent ordinals in HoTT.
- Moral: Define operations on the data structure that is suited for the operation (then transport across with univalence).
- Also: Can prove e.g. transfinite induction.
- Future work: Going beyond ε_0 using a higher inductive type of Brouwer ordinals.

Chuangjie Xu, Fredrik Nordvall Forsberg and Neil Ghani
Three equivalent ordinal notation systems in cubical Agda
CPP 2020.

Conclusions

- Summary: Using mutual definitions and higher inductive types to
- Mo suited for e).
- Als
- Fut type of Bro

Cl ni
Th Agda
CPP 2020.

Thanks!