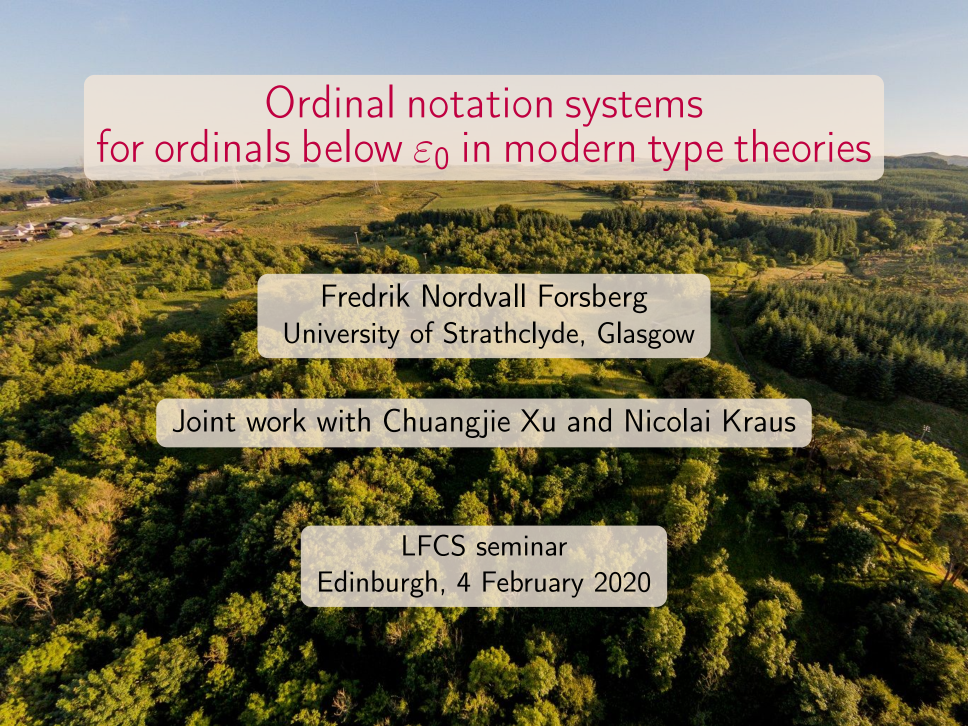


Ordinal notation systems for ordinals below ε_0 in modern type theories

Fredrik Nordvall Forsberg
University of Strathclyde, Glasgow

Joint work with Chuangjie Xu and Nicolai Kraus

LFCS seminar
Edinburgh, 4 February 2020

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

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

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.
- $\omega \cdot 2$, $\omega \cdot 3$, ... 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;

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.

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

Cantor Normal Form

Cantor Normal Form

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

Cantor Normal Form

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

Fact

Every ordinal α can be written uniquely as

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

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

Cantor Normal Form

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

Fact

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 $\beta_1 = \varepsilon_0$.

Cantor Normal Form

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

Fact

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.

Cantor Normal Form

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

Fact

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

$$\alpha > \beta_i > \gamma_j > \dots$$

which must terminate.

Cantor Normal Form

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

Fact

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:

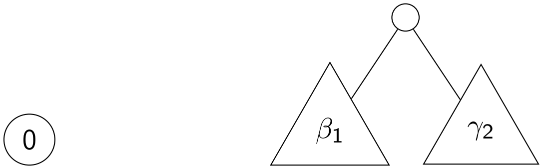
- α is either 0, or
- represented by two ordinals $\alpha = \omega^{\beta_1} + \gamma_2$.

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! [Dershowitz 1993]

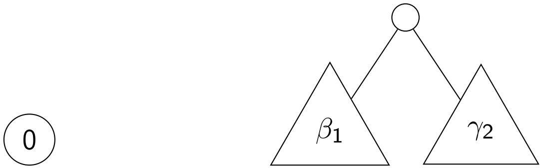


Ordinal notation systems for ordinals below ε_0

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! [Dershowitz 1993]



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

Recovering uniqueness of representation

Three different approaches to recover uniqueness, using features of cubical Agda [Vezzosi, Mörtberg and Abel 2019]:

Recovering uniqueness of representation

Three different approaches to recover uniqueness, using features of cubical Agda [Vezzosi, Mörtberg and Abel 2019]:

- A subset approach
- A mutual approach
- A higher inductive approach

Recovering uniqueness of representation

Three different approaches to recover uniqueness, using features of cubical Agda [Vezzosi, Mörtberg and Abel 2019]:

- A subset approach
- A mutual approach
- A higher inductive approach

Previous work representing ordinals in theorem provers: Manolios and Vroon [2005]; Castéran and Contejean [2006]; Grimm [2013]; Blanchette, Popescu and Traytel [2014]; Blanchette, Fleury and Traytel [2017]; Schmitt [2017]; . . .

Recovering uniqueness of representation

Three different approaches to recover uniqueness, using features of cubical Agda [Vezzosi, Mörtberg and Abel 2019]:

- A subset approach
- A mutual approach
- A higher inductive approach

Previous work representing ordinals in theorem provers: Manolios and Vroon [2005]; Castéran and Contejean [2006]; Grimm [2013]; Blanchette, Popescu and Traytel [2014]; Blanchette, Fleury and Traytel [2017]; Schmitt [2017]; ...

Why care?

Recovering uniqueness of representation

Three different approaches to recover uniqueness, using features of cubical Agda [Vezzosi, Mörtberg and Abel 2019]:

- A subset approach
- A mutual approach
- A higher inductive approach

Previous work representing ordinals in theorem provers: Manolios and Vroon [2005]; Castéran and Contejean [2006]; Grimm [2013]; Blanchette, Popescu and Traytel [2014]; Blanchette, Fleury and Traytel [2017]; Schmitt [2017]; ...

Why care? Unique representatives make the ordinal notations behave like ordinals.

Recovering uniqueness of representation

Three different approaches to recover uniqueness, using features of cubical Agda [Vezzosi, Mörtberg and Abel 2019]:

- A subset approach
- A mutual approach
- A higher inductive approach

Previous work representing ordinals in theorem provers: Manolios and Vroon [2005]; Castéran and Contejean [2006]; Grimm [2013]; Blanchette, Popescu and Traytel [2014]; Blanchette, Fleury and Traytel [2017]; Schmitt [2017]; . . .

Why care? Unique representatives make the ordinal notations behave like ordinals.

Why cubical?

Recovering uniqueness of representation

Three different approaches to recover uniqueness, using features of cubical Agda [Vezzosi, Mörtberg and Abel 2019]:

- A subset approach
- A mutual approach
- A higher inductive approach

Previous work representing ordinals in theorem provers: Manolios and Vroon [2005]; Castéran and Contejean [2006]; Grimm [2013]; Blanchette, Popescu and Traytel [2014]; Blanchette, Fleury and Traytel [2017]; Schmitt [2017]; ...

Why care? Unique representatives make the ordinal notations behave like ordinals.

Why cubical? Want a univalence principle which computes, and higher inductive types.

A subset approach

See e.g. Buchholz [1991]

A subset approach

See e.g. Buchholz [1991]

```
data Tree : Type₀ where
  0 : Tree
  ω^_+_ : Tree → Tree → Tree
```

A subset approach

See e.g. Buchholz [1991]

```
data Tree : Type₀ where
  0 : Tree
  ω^_+_ : Tree → Tree → Tree
```

We single out the trees in Cantor Normal Form:

```
data isCNF : Tree → Type₀ where
  0IsCNF : isCNF 0
  ω^+IsCNF : isCNF a → isCNF b → a ≥ fst b
                → isCNF (ω^ a + b)
```

A subset approach

See e.g. Buchholz [1991]

```
data Tree : Type₀ where
  0 : Tree
  ω^_+_ : Tree → Tree → Tree
```

We single out the trees in Cantor Normal Form:

```
data isCNF : Tree → Type₀ where
  0IsCNF : isCNF 0
  ω^+IsCNF : isCNF a → isCNF b → a ≥ fst b
                  → isCNF (ω^ a + b)
```

This uses _≥_ : Tree→Tree→Type$_0$ (defined inductively), and

```
fst : Tree → Tree
fst 0 = 0
fst (ω^ a + _) = a
```

SigmaOrd

$$\mathsf{SigmaOrd} : \mathsf{Type}_0$$
$$\mathsf{SigmaOrd} = \Sigma \setminus (a : \mathsf{Tree}) \to \mathsf{isCNF}\ a$$

SigmaOrd

$$\mathsf{SigmaOrd} : \mathsf{Type}_0$$
$$\mathsf{SigmaOrd} = \Sigma \; \backslash(a : \mathsf{Tree}) \to \mathsf{isCNF}\ a$$

This is a "subset" of Tree in the sense that isCNF a is proof-irrelevant:

$$\mathsf{isCNFIsPropValued} : \mathsf{isProp}\ (\mathsf{isCNF}\ a)$$

SigmaOrd

$$\mathsf{SigmaOrd} : \mathsf{Type}_0$$
$$\mathsf{SigmaOrd} = \Sigma \ \backslash(a : \mathsf{Tree}) \to \mathsf{isCNF}\ a$$

This is a "subset" of Tree in the sense that isCNF a is proof-irrelevant:

$$\mathsf{isCNFIsPropValued} : \mathsf{isProp}\ (\mathsf{isCNF}\ a)$$

$\boxed{x \equiv y \text{ for any } x, y : \mathsf{isCNF}\ a}$

SigmaOrd

$$\begin{aligned}&\mathsf{SigmaOrd} : \mathsf{Type}_0\\&\mathsf{SigmaOrd} = \Sigma \setminus (a : \mathsf{Tree}) \to \mathsf{isCNF}\ a\end{aligned}$$

This is a "subset" of Tree in the sense that isCNF a is proof-irrelevant:

$$\mathsf{isCNFIsPropValued} : \mathsf{isProp}\ (\mathsf{isCNF}\ a)$$

$x \equiv y$ for any x, y : isCNF a

Pro: Not requiring any fancy features.

Con: "Junk terms". Code duplication.


A mutual approach

Intrinsically Cantor Normal Form ordinals

By using mutual definitions, we get correct-by-construction ordinals in Cantor Normal Form.

Intrinsically Cantor Normal Form ordinals

By using mutual definitions, we get correct-by-construction ordinals in Cantor Normal Form.

We simultaneously define

$$\begin{aligned}
&\mathsf{data}\ \mathsf{MutualOrd} : \mathsf{Type}_0 \\
&\mathsf{data}\ _<_ : \mathsf{MutualOrd} \to \mathsf{MutualOrd} \to \mathsf{Type}_0 \\
&\mathsf{fst} : \mathsf{MutualOrd} \to \mathsf{MutualOrd}
\end{aligned}$$

by induction-induction-recursion [N.-F. 2014].

MutualOrd

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

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

```
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 ]
```

```
fst 0 = 0
fst (ω^ a + _ [ _ ]) = a
```

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 ]
```

```
fst 0 = 0
fst (ω^ a + _ [ _ ]) = a
```

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

Examples

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

Basic properties

Proposition

$\_<\_$ *is proof-irrelevant, i.e.* $p \equiv q$ *for any* $p, q : a < b$*.*

Basic properties

Proposition

$\_<\_$ *is proof-irrelevant, i.e.* $p \equiv q$ *for any* $p, q : a < b$.

Proposition

$\_<\_$ *is trichotomous, i.e. we can define*

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

Basic properties

Proposition

$\_<\_$ *is proof-irrelevant, i.e.* $p \equiv q$ *for any* $p, q : a < b$.

Proposition

$\_<\_$ *is trichotomous, i.e. we can define*

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

Theorem

Transfinite induction holds for MutualOrd, i.e. there is a proof

$$\begin{aligned} MTI &: (P : MutualOrd \to Type\ \ell) \\ &\to (\forall\ x \to (\forall\ y \to y < x \to P\ y) \to P\ x) \\ &\to \forall\ x \to P\ x \end{aligned}$$

Not provable without unique representation!

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 $\omega^\alpha + \omega^\beta = \omega^\beta$.

$\mathbf{0} + b = \{?_0 : \text{MutualOrd}\}$

$a + \mathbf{0} = \{?_1 : \text{MutualOrd}\}$

$(\omega\hat{}\ a + c\,[\,r\,]) + (\omega\hat{}\ b + d\,[\,s\,]) = \{?_2 : \text{MutualOrd}\}$

Addition on MutualOrd

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

$\mathbf{0} + b = b$

$a + \mathbf{0} = \{?_1 : \text{MutualOrd}\}$

$(\omega\text{\^{}}\ a + c\ [\ r\ ]) + (\omega\text{\^{}}\ b + d\ [\ s\ ]) = \{?_2 : \text{MutualOrd}\}$

Addition on MutualOrd

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

$\mathbf{0} + b = b$
$a + \mathbf{0} = a$
$(\omega\hat{}\ a + c\,[\,r\,]) + (\omega\hat{}\ b + d\,[\,s\,]) = \boxed{\{?_2 : \mathrm{MutualOrd}\}}$

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 = {?₂ : MutualOrd}
... | inj₂ a≥b = {?₃ : MutualOrd}
```

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 = {?₃ : MutualOrd}
```

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 ]) [ {?₄ : a≥fst(c+ω^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 ]
```

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

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

Multiplication on MutualOrd

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

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

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

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

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

Cubical Agda allows this via higher inductive types [Lumsdaine and Shulman 2019].

A higher inductive approach

Inspired by Licata's [2014] encoding of finite multisets [Blanchette, Fleury and Traytel 2017] as a HIT

A Higher Inductive Type also allows constructors targetting equalities between elements (and between equalities, equalities between equalities, . . .).

A higher inductive approach

Inspired by Licata's [2014] encoding of finite multisets [Blanchette, Fleury and Traytel 2017] as a HIT

A Higher Inductive Type also allows constructors targetting equalities between elements (and between equalities, equalities between equalities, ...).

Soundness: has a model in cubical sets [Coquand, Huber and Mörtberg 2018].

A higher inductive approach

Inspired by Licata's [2014] encoding of finite multisets [Blanchette, Fleury and Traytel 2017] as a HIT

A Higher Inductive Type also allows constructors targetting equalities between elements (and between equalities, equalities between equalities, . . .).

Soundness: has a model in cubical sets [Coquand, Huber and Mörtberg 2018].

We define:

```
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
```

A higher inductive approach

Inspired by Licata's [2014] encoding of finite multisets [Blanchette, Fleury and Traytel 2017] as a HIT

A Higher Inductive Type also allows constructors targetting equalities between elements (and between equalities, equalities between equalities, . . .).

Soundness: has a model in cubical sets [Coquand, Huber and Mörtberg 2018].

We define:

```
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
```

$p \equiv q$ for all $p, q : a \equiv_{\mathsf{HITOrd}} b$

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 ∎
```

Pattern matching on HITOrd

Pattern matching on HITOrd requires all functions f to respect swap: must show

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

Pattern matching on HITOrd

Pattern matching on HITOrd requires all functions f to respect swap: must show

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

Hence it is convenient to define **commutative** operations on HITOrd.

Pattern matching on HITOrd

Pattern matching on HITOrd requires all functions f to respect swap: must show

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

Hence it is convenient to define **commutative** operations on HITOrd.

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

Hessenberg sum

$$\_\oplus\_ : \mathsf{HITOrd} \to \mathsf{HITOrd} \to \mathsf{HITOrd}$$
$$x \oplus y = \boxed{\{?_0 : \mathsf{HITOrd}\}}$$

Hessenberg sum

$$
\begin{array}{l}
\quad \_\oplus\_ : \mathsf{HITOrd} \to \mathsf{HITOrd} \to \mathsf{HITOrd} \\
\mathbf{0} \qquad\qquad\qquad\quad\; \oplus\, y = \{?_0 : \mathsf{HITOrd}\} \\
(\omega\hat{}\ a \oplus b) \qquad \oplus\, y = \{?_1 : \mathsf{HITOrd}\} \\
(\mathsf{swap}\ a\ b\ c\ i) \oplus y = \{?_2 : \ldots \equiv \ldots\}\ i \\
(\mathsf{trunc}\ p\ q\ i\ j) \oplus y = \{?_3 : \ldots \equiv_{\ldots \equiv \ldots} \ldots\}\ i\ j
\end{array}
$$

Hessenberg sum

$$
\begin{array}{llll}
\multicolumn{4}{l}{\quad \_\oplus\_ : \mathsf{HITOrd} \to \mathsf{HITOrd} \to \mathsf{HITOrd}} \\
\mathbf{0} & \oplus\, y & = & y \\
(\omega\hat{}\ a \oplus b) & \oplus\, y & = & \boxed{\{?_1 : \mathsf{HITOrd}\}} \\
(\mathsf{swap}\ a\ b\ c\ i) & \oplus\, y & = & \boxed{\{?_2 : \ldots \equiv \ldots\}}\ i \\
(\mathsf{trunc}\ p\ q\ i\ j) & \oplus\, y & = & \boxed{\{?_3 : \ldots \equiv_{\ldots \equiv \ldots} \ldots\}}\ i\ j
\end{array}
$$

Hessenberg sum

$$
\begin{aligned}
&\_\oplus\_ : \mathsf{HITOrd} \to \mathsf{HITOrd} \to \mathsf{HITOrd} \\
&\mathbf{0} \quad\quad\quad\quad\quad\; \oplus\ y = y \\
&(\omega\hat{}\ a \oplus b) \quad\; \oplus\ y = \omega\hat{}\ a \oplus (b \oplus y) \\
&(\mathsf{swap}\ a\ b\ c\ i) \oplus y = \boxed{\{?_2 : \ldots \equiv \ldots\}}\ i \\
&(\mathsf{trunc}\ p\ q\ i\ j) \oplus y = \boxed{\{?_3 : \ldots \equiv_{\ldots \equiv \ldots} \ldots\}}\ i\ j
\end{aligned}
$$

Hessenberg sum

$$
\begin{array}{lll}
\multicolumn{3}{l}{\quad \_\oplus\_ : \mathsf{HITOrd} \to \mathsf{HITOrd} \to \mathsf{HITOrd}} \\
\mathbf{0} & \oplus\ y = & y \\
(\omega\hat{}\ a \oplus b) & \oplus\ y = & \omega\hat{}\ a \oplus (b \oplus y) \\
(\mathsf{swap}\ a\ b\ c\ i) & \oplus\ y = & \{?_2 : \ldots \equiv \ldots\}\ i \\
(\mathsf{trunc}\ p\ q\ i\ j) & \oplus\ y = & \{?_3 : \ldots \equiv_{\ldots \equiv \ldots} \ldots\}\ i\ j
\end{array}
$$

In the swap case, we have to prove

$$?_2 : \omega\hat{}\ a \oplus \omega\hat{}\ b \oplus (c \oplus y) \equiv \omega\hat{}\ b \oplus \omega\hat{}\ a \oplus (c \oplus y)$$

Hessenberg sum

$$
\begin{array}{l}
\_\oplus\_ : \mathsf{HITOrd} \to \mathsf{HITOrd} \to \mathsf{HITOrd} \\
\mathbf{0} \qquad\qquad\qquad \oplus\ y = y \\
(\omega\hat{}\ a \oplus b) \qquad \oplus\ y = \omega\hat{}\ a \oplus (b \oplus y) \\
(\mathsf{swap}\ a\ b\ c\ i) \oplus y = \mathsf{swap}\ a\ b\ (c \oplus y)\ \ i \\
(\mathsf{trunc}\ p\ q\ i\ j) \oplus y = \boxed{\{?_3 : \ldots \equiv_{\ldots\equiv\ldots} \ldots\}}\ i\ j
\end{array}
$$

In the swap case, we have to prove

$$?_2 : \omega\hat{}\ a \oplus \omega\hat{}\ b \oplus (c \oplus y) \equiv \omega\hat{}\ b \oplus \omega\hat{}\ a \oplus (c \oplus y)$$

Hessenberg sum

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

In the swap case, we have to prove

$$?_2 : \omega\hat{}\ a \oplus \omega\hat{}\ b \oplus (c \oplus y) \equiv \omega\hat{}\ b \oplus \omega\hat{}\ a \oplus (c \oplus y)$$

Hessenberg sum

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

In the swap case, we have to prove

$$?_2 : \omega\hat{}\ a \oplus \omega\hat{}\ b \oplus (c \oplus y) \equiv \omega\hat{}\ b \oplus \omega\hat{}\ a \oplus (c \oplus y)$$

Proposition

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

Which approach is better?

Which approach is better?

All of them!

Which approach is better?

All of them!

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

Which approach is better?

All of them!

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

Even better:

Theorem
SigmaOrd, MutualOrd and HITOrd are equivalent.

Which approach is better?

All of them!

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

Even better:

Theorem
SigmaOrd, MutualOrd and HITOrd are equivalent.

Using the univalance principle [Voevodsky 2010] (which computes in cubical Agda), equivalent types are identical:

Corollary
SigmaOrd, MutualOrd and HITOrd are identical.

MutualOrd and HITOrd are equivalent

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

MutualOrd and HITOrd are equivalent

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

MutualOrd and HITOrd are equivalent

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

MutualOrd and HITOrd are equivalent

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

$\mathsf{M{\equiv}H : MutualOrd \equiv HITOrd}$

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.

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

Operations via univalence

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

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

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

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

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

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

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

Hence we can construct

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

where

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

is a dependent equality ("path") between $\_<\_$ and $\_<^{\mathsf{H}}\_$.

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

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

for convenience.

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

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.

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

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

Summary and outlook

Conclusions

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

Conclusions

- Summary: Using mutual definitions and higher inductive types to faithfully represent ordinals in cubical Agda.
- 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 cubical Agda.
- Moral: Define operations on the data structure that is suited for the operation (then transport across with univalence).
- 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 cubical Agda.
- Moral: Define operations on the data structure that is suited for the operation (then transport across with univalence).
- 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, New Orleans, USA.

Conclusions

- Su ve types
  to
- Mo suited
  for e).
- Fut type of
  Bro

Cl ni
Th Agda
Cl

Thanks!

References I

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

Jasmin Christian Blanchette, Andrei Popescu, and Dmitriy Traytel.
Cardinals in Isabelle/HOL.
In Gerwin Klein and Ruben Gamboa, editors, *Interactive Theorem Proving*, volume 8558 of *Lecture Notes in Computer Science*, pages 111–127, Heidelberg, Germany, 2014. Springer.

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

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

Thierry Coquand, Simon Huber, and Anders Mörtberg.
On higher inductive types in cubical type theory.
In *Logic in Computer Science*, pages 255–264, New York, USA, 2018. ACM.

Nachum Dershowitz.
Trees, ordinals and termination.
In Marie-Claude Gaudel and Jean-Pierre Jouannaud, editors, *Theory and Practice of Software Development*, volume 668 of *Lecture Notes in Computer Science*, pages 243–250, Heidelberg, Germany, 1993. Springer.

References II

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

Gerhard Hessenberg.
Grundbegriffe der Mengenlehre, volume 1.
Vandenhoeck & Ruprecht, Göttingen, Germany, 1906.

Dan Licata.
What is homotopy type theory?, 2014.
Invited talk at Coq Workshop 2014. Slides available at
`http://dlicata.web.wesleyan.edu/pubs/l14coq/l14coq.pdf`.

Peter Lefanu Lumsdaine and Michael Shulman.
Semantics of higher inductive types.
Mathematical Proceedings of the Cambridge Philosophical Society, pages 1–50, 2019.

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

Fredrik Nordvall Forsberg.
Inductive-inductive definitions.
PhD thesis, Swansea University, 2014.

Peter H. Schmitt.
A mechanizable first-order theory of ordinals.
In Renate Schmidt and Cláudia Nalon, editors, *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 10501 of *Lecture Notes in Computer Science*, pages 331–346, Heidelberg, Germany, 2017. Springer.

References III

Alan Turing.
Checking a large routine.
In *Report of a Conference on High Speed Automatic Calculating Machines*, pages 67–69, Cambridge, UK, 1949. University Mathematical Laboratory.

Andrea Vezzosi, Anders Mörtberg, and Andreas Abel.
Cubical Agda: a dependently typed programming language with univalence and higher inductive types.
Proceedings of the ACM on Programming Languages, 3(ICFP):87:1–87:29, 2019.

Vladimir Voevodsky.
The equivalence axiom and univalent models of type theory.
arXiv 1402.5556, 2010.