A Certified Library of Ordinal Arithmetic

Nicolai Kraus\(^1\), Fredrik Nordvall Forsberg\(^2\), and Chuangjie Xu\(^3\)

\(^1\) University of Nottingham, UK
\(^2\) University of Strathclyde, UK
\(^3\) fortiss GmbH, Germany

Ordinals are a powerful tool for proving termination of processes, for justifying induction and recursion, and generally for many (meta)mathematical constructions. To make use of them in machine-verified proofs, we define a notation system representing ordinals below \(\varepsilon_0\) in the Agda proof assistant, and develop a certified library of ordinal arithmetic.

We work with the typical binary-tree representation of ordinals [2], but implemented in such a way that there are no “junk” terms not denoting ordinals. Binary trees can represent ordinals as follows: the leaf represents 0, and a tree with subtrees representing ordinals \(\alpha\) and \(\beta\) represents the sum \(\omega^\alpha + \beta\). However, an ordinal may have multiple such representations. Using mutual inductive-inductive definitions [4], we define the ordinal notation system simultaneously with an order relation on it. In this way, we recover uniqueness of representation, by insisting that the subtrees are given in a decreasing order. We prove that the notation system allows the principle of transfinite induction and that every term can be classified as a zero, a successor or a limit.

We construct the arithmetic operations including addition, multiplication and exponentiation (with base \(\omega\)), and show that they satisfy the equations/rules in their set-theoretic definitions. For instance, our definition of addition satisfies

\[
\begin{align*}
    a + 0 &= 0 \\
    a + (b + 1) &= a + b + 1 \\
    b \text{ is-lim-of } f \rightarrow c \text{ is-lim-of } (\lambda i. a + f_i) \rightarrow a + b = c
\end{align*}
\]

where \(b \text{ is-lim-of } f\) expresses that \(b\) is the limit of sequence \(f\). Such a relational formulation of correctness is necessary, because our notation system does not have limits constructively. We verify the last rule by defining the inverse operation of addition, i.e. subtraction.

For the commutative Hessenberg addition and multiplication, we work with an equivalent notation system where ordinals below \(\varepsilon_0\) are uniquely represented by finite hereditary multisets. This notation system is defined as a quotient inductive type [1], with a path constructor to identify multiple representations of the same ordinal. This definition is more convenient for constructing the Hessenberg operations. For instance, Hessenberg addition is simply implemented as the concatenation operation on finite multisets. Using the univalence principle [6], we can transport constructions and properties between these two equivalent ordinal notation systems as needed.

The above results have appeared in [5,3], and the Agda development is available at [https://cj-xu.github.io/agda/CertifiedOrdinalArithmetic/](https://cj-xu.github.io/agda/CertifiedOrdinalArithmetic/).
References


