Algorithms for Ordinal Arithmetic

Panagiotis Manolios and Daron Vroon

Georgia Institute of Technology, College of Computing, CERCS Lab
801 Atlantic Drive, Atlanta, Georgia, 30332, USA,
{manolios,vroon}@cc.gatech.edu
http://www.cc.gatech.edu/∼{manolios,vroon}

Abstract. Proofs of termination are essential for establishing the correct behavior of computing systems. There are various ways of establishing termination, but the most general involves the use of ordinals. An example of a theorem proving system in which ordinals are used to prove termination is ACL2. In ACL2, every function defined must be shown to terminate using the ordinals up to ϵ_0. We use a compact notation for the ordinals up to ϵ_0 (exponentially more succinct than the one used by ACL2) and define efficient algorithms for ordinal addition, subtraction, multiplication, and exponentiation. In this paper we describe our notation and algorithms, prove their correctness, and analyze their complexity.

1 Introduction

Termination proofs are of critical importance for the mechanical verification of computing systems. This is the case even with *reactive systems*, non-terminating systems that are engaged in on-going interaction with an environment (*e.g.*, network protocols, operating systems, and distributed databases), as termination proofs are used to show that some desirable behavior is not postponed forever, *i.e.*, to establish liveness properties. Proving termination amounts to showing that a relation is well-founded [1]. Since every well-founded relation can be extended to a total order that is order-isomorphic to an ordinal, it makes sense to base termination proofs on the ordinals.

The theory of the ordinal numbers has been studied extensively for over 100 years and is at the core of Cantor's set theory [4–6]. The ordinals also play a crucial role in logic, *e.g.*, Gentzen proved the consistency of Peano arithmetic using induction up to ϵ_0 [13]. Since Genten's work, proof theorists routinely use ordinals to establish the consistency of various theories. To obtain constructive proofs, constructive ordinals notations are employed [29, 34]. The general theory of ordinal notations was initiated by Church and Kleene [7] and is recounted in Chapter 11 of Roger's book on computability [26].

Ordinal notations are also used in the context of automated reasoning to prove termination. For example, the ordinals up to ϵ_0 are the basis for termination proofs in the ACL2 theorem proving system [15, 16], which has been used on some impressive industrial-scale problems by companies such as AMD, Rockwell

Collins, Motorola, and IBM. ACL2 was used to prove that the floating-point operations performed by the AMD microprocessors are IEEE-754 compliant [25, 28], to analyze bit and cycle accurate models of the Motorola CAP, a digital signal processor [3], and to analyze a model of the JEM1, the world's first silicon JVM (Java Virtual Machine) [14]. Termination proofs have played a key role in various projects that use ACL2 to verify reactive systems. For example, in [20], we develop a theory of refinement for reactive systems that has been used to mechanically verify protocols, pipelined machines, and distributed systems [21, 19, 32]. Ordinals have also played a key role in projects to implement polynomial orderings [23] and multiset relations [27]. The relationship between proof theoretic ordinals and term rewriting is explored in [9, 12].

Even though ordinal notations have been studied and used extensively by various communities for over 100 years, we have not been able to find a comprehensive treatment of arithmetic on ordinal notations. The *ordinal arithmetic problem* for a notational system denoting the ordinals up to some ordinal δ, is as follows: given α and β, expressions in the system denoting ordinals less than δ, is γ the expression corresponding to $\alpha \star \beta$, where $\star$ can be any of $+, -, \cdot$, exponentiation? Solving this problem amounts to defining algorithms for ordinal arithmetic on the notation system in question. We present a solution to the ordinal arithmetic problem for a notational system denoting the ordinals up to ϵ_0. Partial solutions to this problem appear in various books and papers [29, 10, 8, 12, 24, 30, 34], *e.g.*, it is easy to find a definition of $<$ for various ordinal notations, but we have not found any statement of the problem nor any comprehensive solution in previous work.

In this paper, we provide an efficient solution to the ordinal arithmetic problem (for ϵ_0). We use a notational system [2] that is exponentially more succinct than the one used in ACL2 and we give efficient algorithms, whose complexity we analyze. The importance of efficient algorithms becomes apparent when one considers that the definitions are used both to decide ground expressions involving ordinals operations (by computation) and as rewrite rules to simplify expressions involving ordinals, in the general case. In both cases, efficient algorithms can drastically affect the amount of time required to simplify expressions. We have implemented versions of our algorithms in the ACL2 system, mechanically verified their correctness, and developed a library of theorems that can be used to significantly automate reasoning involving the ordinals [22]. Our library substantially increases the extent to which ACL2 can automatically reason about the ordinals, as previously none of the ordinal arithmetic operations were defined and proving termination required defining functions to explicitly construct ordinals. It also allows users to ignore representational issues and instead work in an algebraic setting. As an example, our library will (correctly) reduce

$$7 < m < \omega \ \wedge \ 0 < n < \omega \quad \Rightarrow \quad 5^{(\omega^{\omega})} + \alpha \ < \ [\omega^{n} + w \cdot 3]^{(\omega^{m} \cdot 2 + \omega^{5} \cdot 5)^{\omega}} + \beta$$

to $\alpha < \beta$. Our library is now distributed with the latest version of ACL2 (version 2.7) and has already been used to give a constructive proof of Dickson's lemma [33]. Knowledge of ACL2 is not required to read this paper, as only the algorithms and their complexity are discussed; a discussion of the ACL2 library will appear elsewhere [22].

The paper is organized as follows. In Section 2 we give an overview of the ordinals and describe our representation. In Section 3 we define the ordinal arithmetic operations and analyze their complexity. In Section 4 we prove that our algorithms are correct. We analyze the complexity of the algorithms before proving correctness because our algorithms are driven by complexity considerations. Due to space limitations, some of the proofs in this paper have been elided or turned into proof sketches. Finally, in Section 5 we conclude and outline future work.

2 Ordinals

We review the theory of ordinals [11, 18, 29]. A relation $\prec$ is *well-founded* if every $\prec$-decreasing sequence is finite. A *woset* is a pair $\langle X, \prec \rangle$, where X is a set, and $\prec$ is a *well-ordering*, a total, well-founded relation, over X. Given a woset, $\langle X, \prec \rangle$, and an element $a \in X$, X_a is defined to be $\{x \in X \mid x \prec a\}$. An *ordinal* is a woset $\langle X, \prec \rangle$ such that for all $a \in X$, $a = X_a$. It follows that if $\langle X, \prec \rangle$ is an ordinal and $a \in X$, then a is an ordinal and that $\prec$ is equivalent to $\in$. In the sequel, we will use lower case greek letters to denote ordinals and $<$ or $\in$ to denote the ordering.

Given two wosets, $\langle X, \prec \rangle$ and $\langle X', \prec' \rangle$, a function $f : X \rightarrow X'$ is said to be an *order-isomorphism* if it is a bijection and for all $x, y \in X$, $x \prec y$ iff $f.x \prec' f.y$. Two wosets are said to be *order-isomorphic* if there exists an order-isomorphism between them. A basic result of set theory states that every woset is order-isomorphic to a unique ordinal. Given a woset $\langle X, \prec \rangle$, we will denote the ordinal to which it is order-isomorphic as $Ord(X, \prec)$. Since every well-founded relation can be extended to a woset, we see that the theory of the ordinals is the most general setting for proving termination.

Given an ordinal, α, we define its *successor*, denoted α' to be $\alpha \cup \{\alpha\}$. There is clearly a minimal ordinal, $\emptyset$. It is commonly denoted by 0. The next smallest ordinal is $0' = \{0\}$ and is denoted by 1. The next is $1' = \{0, 1\}$ and is denoted by 2. Continuing in this manner, we get all the natural numbers. A *limit ordinal* is an ordinal > 0 that is not a successor. The set of natural numbers, denoted ω, is the smallest limit ordinal.

2.1 Ordinal Arithmetic

We define addition, subtraction, multiplication, and exponentiation for the ordinals. After each definition, we list various properties; proofs can be found in texts on set theory [11, 18, 29].

Definition 1. *$\alpha + \beta = Ord(A, <_A)$ where $A = (\{0\} \times \alpha) \cup (\{1\} \times \beta)$ and $<_A$ is the lexicographic ordering on A.*

Ordinal addition satisfies the following properties.

$$
\begin{array}{ll}
\alpha + 1 \;=\; \alpha' & \\
(\alpha+\beta)+\gamma \;=\; \alpha+(\beta+\gamma) & \text{(associativity)} \\
(\beta<\gamma) \;\Rightarrow\; \alpha+\beta<\alpha+\gamma & \text{(strict right monotonicity)} \\
(\beta<\gamma) \;\Rightarrow\; \beta+\alpha\leq\gamma+\alpha & \text{(weak left monotonicity)} \\
(\alpha<\omega^{\beta}) \;\Rightarrow\; \alpha+\omega^{\beta}=\omega^{\beta} & \text{(additive principal property)} \\
(\alpha,\beta<\omega^{\gamma}) \;\Rightarrow\; \alpha+\beta<\omega^{\gamma} & \text{(closure of additive principal ordinals)}
\end{array}
$$

Note that addition is not commutative, *e.g.*, $1+\omega=\omega<\omega+1$.

Definition 2. *$\alpha-\beta$ is defined to be 0 if $\alpha\leq\beta$, otherwise, it is the unique ordinal, ξ such that $\beta+\xi=\alpha$.*

Definition 3. *$\alpha\cdot\beta = Ord(A,<_A)$ where $A=\bigcup_{\xi<\beta}(\{\xi\}\times\alpha)$ and $<_A$ is the lexicographic ordering on A.*

Ordinal multiplication satisfies the following properties.

$$
\begin{array}{ll}
n\in\omega\wedge n>0 \;\Rightarrow\; n\cdot\omega=\omega & \\
(\alpha\cdot\beta)\cdot\gamma \;=\; \alpha\cdot(\beta\cdot\gamma) & \text{(associativity)} \\
(\beta<\gamma) \;\Rightarrow\; \alpha\cdot\beta<\alpha\cdot\gamma & \text{(strict right monotonicity)} \\
(\beta<\gamma) \;\Rightarrow\; \beta\cdot\alpha\leq\gamma\cdot\alpha & \text{(weak left monotonicity)} \\
\alpha\cdot(\beta+\gamma) \;=\; (\alpha\cdot\beta)+(\alpha\cdot\gamma) & \text{(left distributivity)}
\end{array}
$$

Note that commutivity and right distributivity do not hold for multiplication, *e.g.*, $2\cdot\omega \;=\; \omega \;<\; \omega\cdot 2$, and $(\omega+1)\cdot\omega \;=\; \omega\cdot\omega \;<\; \omega\cdot\omega+\omega$.

Definition 4. *Given any ordinal, α, we define exponentiation using transfinite induction: $\alpha^0=1$, $\alpha^{\beta+1}=\alpha\cdot\alpha^{\beta}$, and for β a limit ordinal, $\alpha^{\beta}=\bigcup_{\xi<\beta}\alpha^{\xi}$.*

Ordinal exponentiation satisfies the following properties.

$$
\begin{array}{ll}
\alpha^{\beta}\cdot\alpha^{\gamma} \;=\; \alpha^{\beta+\gamma} & \\
(\alpha^{\beta})^{\gamma} \;=\; \alpha^{\beta\cdot\gamma} & \\
(\beta<\gamma) \;\Rightarrow\; \alpha^{\beta}<\alpha^{\gamma} & \text{(strict right monotonicity)} \\
(\beta<\gamma) \;\Rightarrow\; \beta^{\alpha}\leq\gamma^{\alpha} & \text{(weak left monotonicity)} \\
(p\in\omega) \;\Rightarrow\; p^{\omega}=\omega &
\end{array}
$$

2.2 Ordinal Notations

Using the ordinal operations, we can construct a hierarchy of ordinals: $0,1,2,\ldots,\omega,\omega+1,\omega+2,\ldots,\omega\cdot2,\omega\cdot2+1,\ldots,\omega^2,\ldots,\omega^3,\ldots,\omega^{\omega},\ldots$, and so on. The ordinal $\omega^{\omega^{\omega^{\cdots}}}$ is called ϵ_0, and it is the smallest ordinal, α, for which $\omega^{\alpha}=\alpha$; such ordinals are called ϵ-ordinals. We consider notations for the ordinals less than ϵ_0; the notations are based on the Cantor Normal Form for ordinals [29].

Theorem 1. *For every ordinal $\alpha\neq0$, there are unique $\alpha_1\geq\alpha_2\geq\cdots\geq\alpha_n (n\geq 1)$ such that $\alpha=\omega^{\alpha_1}+\cdots+\omega^{\alpha_n}$.*

For every $\alpha \in \epsilon_0$, we have that $\alpha < \omega^\alpha$, as ϵ_0 is the smallest ϵ-ordinal. Thus, we can add the restriction that $\alpha > \alpha_1$ for these ordinals. This is essentially the ordinal notation used in ACL2 [15, 16]. However, as $\omega^\alpha \cdot k + \omega^\alpha = \omega^\alpha \cdot (k+1)$ and $n \in \omega$, we can collect like terms and rewrite the normal form as follows [2].

Corollary 1. (Cantor Normal Form) *For every ordinal $\alpha \in \epsilon_0$, there are unique $n, p \in \omega, \alpha_1 > \cdots > \alpha_n > 0$, and $x_1, \ldots, x_n \in \omega \backslash \{0\}$ such that $\alpha > \alpha_1$ and $\alpha = \omega^{\alpha_1} x_1 + \cdots + \omega^{\alpha_n} x_n + p$.*

By the *size* of an ordinal under a representation, we mean the number of bits needed to denote the ordinal in that representation.

Lemma 1. *The ordinal representation in Corollary 1 is exponentially more succinct than the representation in Theorem 1.*

Proof Consider $\omega \cdot k$: it requires $O(k)$ bits with the representation in Theorem 1 and $O(\log k)$ bits with the representation in Corollary 1. $\square$

The ordinal notation we employ is based on Corollary 1 and we now describe it in detail. We use a definition of lists based on the notion of dotted pairs as defined in the Lisp programming language. Nested pairs of the form $\langle a_1, \langle a_2, \ldots \langle a_{n-1}, a_n \rangle \ldots \rangle \rangle$ are lists and can be abbreviated by replacing ",$\langle \ldots \rangle$" by "$\ldots$" and "$\langle \ldots \rangle$" by "$(\ldots)$", *e.g.*, $\langle a_1, \langle a_2, \langle a_3, a_4 \rangle \rangle \rangle$ can be written as $(a_1\ a_2\ a_3\ ,\ a_4)$. We will also use `nil` to denote the end of a list. CNF(α) denotes our representation of the ordinal α. If $\alpha = \sum_{i=1}^{n} \omega^{\alpha_i} x_i + p$ then CNF(α) = $(\langle \mathrm{CNF}(\alpha_1), x_1 \rangle\ \langle \mathrm{CNF}(\alpha_2), x_2 \rangle\ \ldots\ \langle \mathrm{CNF}(\alpha_n), x_n \rangle\ ,\ p)$.

We now define some basic functions for manipulating lists. Some of our functions are partial, *i.e.*, they are not specified for all inputs. In such cases, we never use them outside of their intended domain. **first**(⟨a,b⟩) returns `a` and **rest**(⟨a,b⟩) returns `b`. "`a . b`" abbreviates ⟨a,b⟩. When a sequence of < *condition* > : < *result* > forms are used to define a function, the conditions should be read from top to bottom until a condition that holds is found and then the corresponding result is returned.

```
firstn(a,n) ;n is a natural number
  n = 0 : nil
  true  : first(a) . firstn(rest(a),n-1)

restn(a,n) ;n is a natural number
  n = 0 : a
  true  : restn(rest(a),n-1)

fe(a) ;the first exponent of a
  atom(a) : 0
  true    : first(first(a))

fc(a) ;the first coefficient of a
  atom(a) : a
  true    : rest(first(a))

atom(a) ;true iff a is not a list
  a is not a list : true
  true            : false

|a| ;the length of a
  atom(a) : 0
  true    : 1 + |rest(a)|

#a ;the size of a
  atom(a) : 1
  true    : #fe(a) + #rest(a)

a@b ;append operator
  atom(a) : b
  true    : first(a) . rest(a)@b
```

Note that `a@b` takes time |a| to complete.

3 Definitions and Complexity

We refer to the usual arithmetic operations on integers as $<_\omega$, $+_\omega$, $-_\omega$, $\cdot_\omega$, and exp_ω. For our analysis, we assume that these operations have constant running time. One can later account for the integer operations by using the fastest known algorithms. This approach allows us to focus on the interesting aspects of our algorithms, namely the aspects pertaining to the ordinal representations.

$\mathbf{cmp}_\omega$`(p,q)` ;*ordering on naturals*

`p` $<_\omega$ `q`	: lt
`q` $<_\omega$ `p`	: gt
`true`	: eq

$\mathbf{cmp_o}$`(a,b)` ;*ordering on ordinals*

atom`(a)` $\wedge$ **atom**`(b)`	: $\mathbf{cmp}_\omega$`(a,b)`
atom`(a)`	: lt
atom`(b)`	: gt
$\mathbf{cmp_o}$`(`**fe**`(a),`**fe**`(b))` $\neq eq$	: $\mathbf{cmp_o}$`(`**fe**`(a),`**fe**`(b))`
$\mathbf{cmp}_\omega$`(`**fc**`(a),`**fc**`(b))` $\neq eq$	: $\mathbf{cmp}_\omega$`(`**fc**`(a),`**fc**`(b))`
`true`	: $\mathbf{cmp_o}$`(`**rest**`(a),`**rest**`(b))`

`a` $<_o$ `b` ;< *for ordinals*

$\mathbf{cmp_o}$`(a,b) =` lt	: `true`
`true`	: `false`

Theorem 2. $\mathbf{cmp_o}(\mathtt{a},\mathtt{b})$ *runs in time* $O(\min(\#\mathtt{a},\#\mathtt{b}))$.

Proof Easily proven by induction. Note that `a` and `b` are explored together and when an atom is encountered, the other is not explored further. $\square$

cnfp`(a)` ;*ordinal recognizer*

atom`(a)` : `a` $\in \omega$
`true` : $\neg$**atom**`(`**first**`(a))`
$\wedge$ **fc**`(a)` $\in \omega$
$\wedge$ `0` $<_\omega$ **fc**`(a)`
$\wedge$ **cnfp**`(`**fe**`(a))`
$\wedge$ **cnfp**`(`**rest**`(a))`
$\wedge$ **fe**`(`**rest**`(a))` $<_o$ **fe**`(a)`

Lemma 2. $\langle \forall x \in \omega :: \binom{2x}{x} \geq 2^x \rangle$.

Proof If $x = 0$, then $1 \geq 1$, else $\binom{2x}{x} = \frac{(2x)!}{x!x!} = \frac{(2x)(2x-1)\cdots(x+1)}{x(x-1)\cdots 1} \geq 2 \cdots 2 \geq 2^x$ $\square$

Lemma 3. $\langle \forall x, y \in \omega :: x \leq y \quad \Rightarrow \quad x^x y^y 2^x \leq (x+y)^{x+y} \rangle$

Proof $(x+y)^{x+y} \geq$ { by the binomial theorem } $x^x y^y \binom{x+y}{x} \geq \{y \geq x\}$ $x^x y^y \binom{2x}{x} \geq$ { by Lemma 2 } $x^x y^y 2^x$ $\square$

Lemma 4. $\langle \forall x, y \in \omega :: x \leq y \quad \Rightarrow \quad x \log x + y \log y + x \leq (x+y)\log(x+y) \rangle$

Proof By Lemma 3, we know that $x^x y^y 2^x \le (x+y)^{x+y}$. By the monotonicity of log, this implies $\log(x^x y^y 2^x) \le \log[(x+y)^{x+y}]$, *i.e.*, $\log x^x + \log y^y + \log 2^x \le (x+y)\log(x+y)$, *i.e.*, $x \log x + y \log y + x \le (x+y)\log(x+y)$. $\square$

Theorem 3. **cnfp**(a) *runs in time* $O(\#\mathtt{a}(\log \#\mathtt{a}))$.

Proof The running time can be given by the (non-linear) recurrence relation

$$T(\mathtt{a}) = \begin{cases} k, & \text{if } \mathbf{atom}(\mathtt{a}) \\ T(\mathbf{fe}(\mathtt{a})) + T(\mathbf{rest}(\mathtt{a})) + \min(\#\mathbf{fe}(\mathtt{a}), \#\mathbf{rest}(\mathtt{a})) + k, & \text{otherwise} \end{cases}$$

for some constant, k, by Theorem 2. We will show by induction on $\#\mathtt{a}$, that $T(\mathtt{a}) \le c(\#\mathtt{a})(\log \#\mathtt{a}) + t$ where c, t are constants such that $t \ge k$ and $c \ge 3t$. In the base case, we have $T(\mathtt{a}) \le k \le t$. For the induction step, let $x = \min(\#\mathbf{fe}(\mathtt{a}), \#\mathbf{rest}(\mathtt{a}))$ and $y = \max(\#\mathbf{fe}(\mathtt{a}), \#\mathbf{rest}(\mathtt{a}))$. Note that $x + y = \#\mathtt{a}$. We have: $T(\mathtt{a}) = T(\mathbf{fe}(\mathtt{a})) + T(\mathbf{rest}(\mathtt{a})) + x + k \le \{$ Induction hypothesis $\}$ $cx\log x + t + cy \log y + t + x + k \le \{cx \ge 2t + x \text{ as } c \ge 3t\}$ $c(x \log x + y \log y + x) + k \le \{$ Lemma 4 $, t \ge k\}$ $c(\#\mathtt{a})(\log \#\mathtt{a}) + t$. $\square$

```
a +o b ;ordinal addition
  atom(a)  ∧  atom(b)  : a +ω b
  cmpo(fe(a),fe(b)) = lt : b
  cmpo(fe(a),fe(b)) = eq : ⟨fe(a),fc(a) +ω fc(b)⟩ . rest(b)
  true                   : ⟨fe(a),fc(a)⟩ . (rest(a) +o b)
```

Theorem 4. $\mathtt{a} +_o \mathtt{b}$ *runs in time* $O(\min(\#\mathtt{a}, |\mathtt{a}|\#\mathbf{fe}(\mathtt{b})))$.

Proof In the worst case, we compare all the exponents in a to **fe**(b), thus, by Theorem 2, #a is an upper bound. In this same case we are comparing **fe**(b) |a| times, thus, again by Theorem 2, |a|#**fe**(b) is an upper bound. $\square$

```
a -o b ;ordinal subtraction
  atom(a) ∧  atom(b)  ∧  a ≤ω b : 0
  atom(a) ∧  atom(b)            : a -ω b
  cmpo(fe(a),fe(b)) = lt        : 0
  cmpo(fe(a),fe(b)) = gt        : a
  fc(a) <ω fc(b)                : 0
  fc(a) >ω fc(b)                : ⟨fe(a),fc(a) -ω fc(b)⟩ . rest(a)
  true                          : rest(a) -o rest(b)
```

Theorem 5. $\mathtt{a} -_o \mathtt{b}$ *runs in time* $O(\min(\#\mathtt{a}, \#\mathtt{b}))$.

Proof In the worst case, we recur along both a and b, as with $\mathbf{cmp_o}$. $\square$

Here is a first attempt at defining multiplication:

```
a *o b ;ordinal multiplication
  a = 0   ∨  b = 0     : 0
  atom(a)  ∧  atom(b) : a ·ω b
  atom(b)             : ⟨fe(a),fc(a) ·ω b⟩ . rest(a)
  true                : ⟨fe(a) +o fe(b),fc(b)⟩ . (a *o rest(b))
```

The problem with this definition is its running time. In the `true` case, the algorithm walks down b, adding **fe**(a) to each exponent of b. This is equivalent to adding some ordinal, cto a decreasing sequence of ordinals $\{\mathtt{d}_1, \mathtt{d}_2, \ldots, \mathtt{d}_n\}$. Using the addition algorithm, we find that for each d_i, $\mathbf{fe}(\mathtt{d}_i)$ is compared to each exponent of cuntil the first exponent of csuch that $\mathbf{fe}(\mathtt{d}_i)$ is $\geq$ this exponent is found. But since the $\mathtt{d}_i$'s are decreasing, we know that $\mathbf{fe}(\mathtt{d}_i) \geq \mathbf{fe}(\mathtt{d}_{i+1})$. Therefore, if the jth exponent of cis greater than $\mathbf{fe}(\mathtt{d}_i)$, we know that it is greater than $\mathbf{fe}(\mathtt{d}_{i+1})$. This means that simply adding each element of the decreasing sequence to cis inefficient. If we can keep track of how many exponents of cwe went through before adding $\mathtt{d}_i$, we can just skip over those when we add $\mathtt{d}_{i+1}$. Thus, a quicker way to compute multiplication is to use the following functions.

```
c(a,b) ;finds the index of the first exponent of a that is ≤ fe(b)
  fe(b) <_o fe(a) : 1 +_ω c(rest(a),b)
  true            : 0

count(a,b,n) ;skips over the first n elements of a and then calls c
  true : n + c(restn(a,n),b)

padd(a, b, n) ;skips over the first n elements of a and then adds the rest to b
  true : firstn(a,n)@(restn(a,n) +_o b)

pmult(a,b,n) ;psuedo multiplication for ordinals
  a = 0  ∨  b = 0       : 0
  atom(a)  ∧  atom(b)   : a ·_ω b
  atom(b)               : ⟨fe(a),fc(a) ·_ω b⟩ . rest(a)
  true                  : ⟨padd(fe(a),fe(b),m),fc(b)⟩
                            . (pmult(a,rest(b),m))
                              where m = count(fe(a),fe(b),n)

a ·_o b ;smarter ordinal multiplication
  true : pmult(a,b,0)
```

For the next few theorems and lemmas, let $\mathtt{a} = (\langle \mathtt{a}_1, \mathtt{x}_1\rangle\langle \mathtt{a}_2, \mathtt{x}_2\rangle \ldots \langle \mathtt{a}_\mathtt{n}, \mathtt{x}_\mathtt{n}\rangle, \mathtt{p})$, $\mathtt{a}_1 = (\langle \mathtt{d}_1, \mathtt{z}_1\rangle\langle \mathtt{d}_2, \mathtt{z}_2\rangle \ldots \langle \mathtt{d}_\mathtt{k}, \mathtt{z}_\mathtt{k}\rangle, \mathtt{r})$, and $\mathtt{b} = (\langle \mathtt{b}_1, \mathtt{y}_1\rangle\langle \mathtt{b}_2, \mathtt{y}_2\rangle \ldots \langle \mathtt{b}_\mathtt{m}, \mathtt{y}_\mathtt{m}\rangle, \mathtt{q})$.

Lemma 5. $\mathbf{c}(\mathtt{a}, \mathtt{b})$ *takes time* $O(\sum_{i=1}^{\mathbf{c}(\mathtt{a},\mathtt{b})+1} \min(\#\mathtt{a}_i, \#\mathbf{fe}(b)))$.

Lemma 6. $\mathbf{count}(\mathtt{a}, \mathtt{b}, \mathtt{s})$ *takes time* $O(\mathtt{s} + \sum_{i=\mathtt{s}+1}^{\mathbf{c}(\mathtt{a},\mathtt{b})+1} \min(\#\mathtt{a}_i, \#\mathbf{fe}(\mathtt{b})))$.

Lemma 7. $\mathbf{padd}(\mathtt{a}, \mathtt{b}, \mathtt{s})$ *runs in time* $O(\min(\#\mathbf{fe}(\mathbf{restn}(\mathtt{a}, \mathtt{s})), \#\mathbf{fe}(\mathtt{b})) + \mathtt{s})$ *when* $\mathtt{s} \geq \mathbf{c}(\mathtt{a}, \mathtt{b})$.

Lemma 8. $\mathtt{d} < \mathtt{b} \quad \Rightarrow \quad \mathbf{c}(\mathtt{a}, \mathtt{b}) \leq \mathbf{c}(\mathtt{a}, \mathtt{d})$

Lemma 9. $\mathtt{k} \leq \mathbf{c}(\mathtt{a}, \mathtt{b}) \quad \Rightarrow \quad \mathbf{c}(\mathtt{a}, \mathtt{b}) = \mathbf{count}(\mathtt{a}, \mathtt{b}, \mathtt{k})$.

Theorem 6. **pmult***(a,b,s) runs in time* $O(|\mathbf{fe}(\mathtt{a})||\mathtt{b}| + \#\mathbf{restn}(\mathbf{fe}(\mathtt{a}), \mathtt{s}) + \#\mathtt{b})$ *if* $\mathtt{s} \leq \mathbf{c}(\mathbf{fe}(\mathtt{a}), \mathbf{fe}(\mathtt{b}))$.

Proof Let $\mathtt{m} = \mathbf{count}(\mathbf{fe}(\mathtt{a}), \mathbf{fe}(\mathtt{b}), \mathtt{s})$; then $\mathtt{m} = \mathbf{c}(\mathbf{fe}(\mathtt{a}), \mathbf{fe}(\mathtt{b}))$ by Lemma 9. Thus, using Lemmas 6 and 7, we can construct the following recurrence relation to bound the running time of **pmult**:

$$T(\mathtt{a},\mathtt{b},\mathtt{s}) = \begin{cases} d, \quad \text{if } \mathbf{atom}(\mathtt{b}) \quad \vee \quad \mathtt{a}{=}\ 0 \\ T(\mathtt{a}, \mathbf{rest}(\mathtt{b}), \mathtt{m}) + k_1 \cdot (\mathtt{s} + \sum_{i=\mathtt{s}+1}^{\mathtt{m}+1} \min(\#\mathtt{d}_i, \#\mathbf{fe}(\mathbf{fe}(\mathtt{b})))) \\ \quad +k_2 \cdot (\min(\#\mathbf{fe}(\mathbf{restn}(\mathbf{fe}(\mathtt{a}), \mathtt{m})), \#\mathbf{fe}(\mathtt{b})) + \mathtt{m}) + d, \quad \text{otherwise} \end{cases}$$

for some constants k_1, k_2, and d. The proof involves using induction to show that $T(\mathtt{a},\mathtt{b},\mathtt{s}) \leq k \cdot (|\mathbf{fe}(\mathtt{a})||\mathtt{b}| + \#\mathbf{restn}(\mathbf{fe}(\mathtt{a}), \mathtt{s}) + \#\mathtt{b})$ where $k \geq k_1 + k_2 + d$. $\square$

Corollary 2. $\mathtt{a} \cdot_o \mathtt{b}$ *runs in time* $O(|\mathbf{fe}(\mathtt{a})||\mathtt{b}| + \#\mathbf{fe}(\mathtt{a}) + \#\mathtt{b})$.

```
exp₁(p,b) ;raising a natural number to an infinite ordinal power
  cmpₒ(fe(b),1) = eq : (⟨fc(b),exp_ω(p,rest(b))⟩,0)
  atom(rest(b))      : (⟨(⟨fe(b) −ₒ 1,fc(b)⟩,0),exp_ω(p,rest(b))⟩,0)
  true               : (⟨⟨fe(b) −ₒ 1,1⟩ . fe(c),fc(c)⟩,0)
                         where c = exp₁(p,rest(b))
```

Theorem 7. $\mathbf{exp_1}$ *runs in time* $O(|b|)$

```
exp₂(a,q) ;raising a limit ordinal to a positive integer power
  q = 1 : a
  true  : (⟨fe(a) ·ₒ (q - 1),1⟩,0) ·ₒ a
```

Theorem 8. $\mathbf{exp_2}(\mathtt{a},\mathtt{q})$ *runs in time* $O(|\mathbf{fe}(\mathtt{a})||\mathtt{a}| + \#\mathtt{a})$

Proof Note that $\mathbf{fe}(\mathtt{a}) \cdot_o (q-1)$ takes constant time, since $\mathtt{q}-1$ is of size 1. Also, note that the $\#(\mathbf{fe}(\mathtt{a}) \cdot_o (\mathtt{q}-1)) = \#\mathbf{fe}(\mathtt{a})$ and $|\mathbf{fe}(\mathtt{a}) \cdot_o (\mathtt{q}-1)| = |\mathbf{fe}(\mathtt{a})|$. So, by Corollary 2, we get that the running time is $k \cdot (|\mathbf{fe}(\mathtt{a})||\mathtt{a}| + \#\mathbf{fe}(\mathtt{a}) + \#\mathtt{a}) \leq k \cdot (|\mathbf{fe}(\mathtt{a})||\mathtt{a}| + \#\mathtt{a} + \#\mathtt{a}) \leq 2k \cdot (|\mathbf{fe}(\mathtt{a})||\mathtt{a}| + \#\mathtt{a})$ for some constant, k. $\square$

```
limitp(a) ;returns true if a = 0 or a represents a limit ordinal
  atom(a) : a = 0
  true    : limitp(rest(a))

limitpart(a) ;returns the greatest ordinal, b, such that limitp(b) and b <ₒ a
  atom(a) : 0
  true    : first(a) . limitpart(rest(a))

natpart(a) ;returns the natural part of an ordinal
  atom(a) : a
  true    : natpart(rest(a))

helper-exp₃(a,p,n,q) ;helper function for exp₃
  q = 0 : p
  true  : padd(exp₂(a,q) ·ₒ p, helper-exp₃(a,p,n,q-1), n)

exp₃(a,q) ;raising an infinite ordinal to a natural power
  q = 0     : 1
  q = 1     : a
  limitp(a) : exp₂(a,q)
  true      : padd(firstn(exp₂(c,q), helper-exp₃(c,natpart(a),n,q-1), n))
              where c = limitpart(a) and n = |a|
```

Lemma 10. $\mathbf{helper\text{-}exp_3}(\mathtt{a},\mathtt{p},\mathtt{n},\mathtt{q})$ *runs in time* $O(\mathtt{q}\cdot(|\mathbf{fe}(\mathtt{a})||\mathtt{a}|+\#\mathtt{a}))$ *when* $\mathbf{limitp}(\mathtt{a})$*,* $\mathtt{p}\in\omega$*,* $\mathtt{n}=|\mathtt{a}|$*, and* $\mathtt{q}$ *is a positive natural number.*

Theorem 9. $\mathbf{exp_3}(\mathtt{a},\mathtt{q})$ *runs in time* $O(\mathtt{q}\cdot(|\mathbf{fe}(\mathtt{a})||\mathtt{a}|+\#\mathtt{a}))$.

```
exp4(a,b) ;raising an infinite ordinal to a possibly infinite power
  true : (⟨fe(a) ·o limitpart(b),1⟩,0) ·o exp3(a,natpart(b))
```

Lemma 11. $\mathtt{k}\in\omega \quad \Rightarrow \quad \#\mathbf{padd}(\mathtt{a},\mathtt{b},\mathtt{k})\leq\#\mathtt{a}+\#\mathtt{b}$.

Lemma 12. $\mathbf{limitp}(\mathtt{b}) \;\Rightarrow\; |\mathtt{a}\cdot_o\mathtt{b}|=|\mathtt{b}|$.

Lemma 13. $\mathbf{limitp}(\mathtt{b}) \;\Rightarrow\; \#(\mathtt{a}\cdot_o\mathtt{b})\leq\#\mathbf{fe}(\mathtt{a})|\mathtt{b}|+\#\mathtt{b}$.

Theorem 10. $\mathbf{exp_4}(\mathtt{a},\mathtt{b})$ *runs in time* $O(\mathbf{natpart}(\mathtt{b})[|\mathtt{a}||\mathtt{b}|+|\mathbf{fe}(\mathtt{a})||\mathtt{a}|+\#\mathtt{a}]+\#\mathbf{fe}(\mathbf{fe}(\mathtt{a}))|\mathtt{b}|+\#\mathtt{b})$.

Proof The proof involves analyzing the complexity of the two multiplications and the call to $\mathbf{exp_3}$, and uses Lemmas 12 and 13 to compute the length and size of $\mathbf{fe}(\mathtt{a})\cdot_o\mathbf{limitpart}(\mathtt{b})$. $\square$

```
expo(a,b) ;ordinal exponentiation (raises a to the b power)
  b = 0    ∨  a = 1      : 1
  a = 0                  : 0
  atom(a)  ∧  atom(b)    : expω(a,b)
  atom(a)                : exp1(a,b)
  atom(b)                : exp3(a,b)
  true                   : exp4(a,b)
```

Theorem 11. $\mathbf{exp_o}(\mathtt{a},\mathtt{b})$ *runs in time* $O(\mathbf{natpart}(\mathtt{b})[|\mathtt{a}||\mathtt{b}|+|\mathbf{fe}(\mathtt{a})||\mathtt{a}|+\#\mathtt{a}]+\#\mathbf{fe}(\mathbf{fe}(\mathtt{a}))|\mathtt{b}|+\#\mathtt{b})$.

Proof This follows directly from Theorems 7, 9, and 10. $\square$

4 Proofs Of Correctness

In this section we prove the correctness of the algorithms for ordinal arithmetic given in the previous section. In the sequel, the ordinals α and β have the following Cantor normal form decompositions $\alpha=\sum_{i=1}^{n}\omega^{\alpha_i}x_i+p$ and $\beta=\sum_{i=1}^{m}\omega^{\beta_i}y_i+q$. Also, we will let $\mathtt{a}=(\langle\mathtt{a}_1,\mathtt{x}_1\rangle\ldots\langle\mathtt{a}_n,\mathtt{x}_n\rangle\ ,\ \mathtt{p})=\mathrm{CNF}(\alpha)$, and $\mathtt{b}=(\langle\mathtt{b}_1,\mathtt{y}_1\rangle\ldots\langle\mathtt{b}_m,\mathtt{y}_m\rangle\ ,\ \mathtt{q})=\mathrm{CNF}(\beta)$. We start with a comparison function for ordinals: *cmp*(α,β) is *lt* if $\alpha<\beta$, *gt* if $\alpha>\beta$, and *eq* otherwise.

Theorem 12. *For all* $\alpha,\beta\in\epsilon_0$*,* $\mathbf{cmp_o}(\mathrm{CNF}(\alpha),\mathrm{CNF}(\beta))=\mathit{cmp}(\alpha,\beta)$.

Proof The proof is by induction on the sizes of $\mathrm{CNF}(\alpha)$ and $\mathrm{CNF}(\beta)$. Only a sketch is given here. The correctness is trivial if $\alpha\in\omega$ or $\beta\in\omega$. Otherwise, the key insight is that $\sum_{i=2}^{n}\omega^{\alpha_i}x_i<\omega^{\alpha_1}$. If $\alpha_1\neq\beta_1$, then suppose without loss of generality that $\alpha_1<\beta_1$. Then $\alpha<\omega^{\alpha_1}(x_1+1)<\omega^{\beta_1}\leq\beta$.

By the inductive hypothesis, $\mathbf{cmp_o}(\mathbf{fe}(\mathtt{a}), \mathbf{fe}(\mathtt{b})) = lt$, so $\mathbf{cmp_o}$ is correct. If $\alpha_1 = \beta_1$ and $x_1 \neq y_1$, then without loss of generality, suppose $x_1 < y_1$. Then $\alpha < \omega^{\alpha_1}(x_1 + 1) \leq \omega^{\beta_1} y_1 \leq \beta$. Hence, by the correctness of $\mathbf{cmp}_\omega$, $\mathbf{cmp_o}$ is correct. Otherwise, $\omega^{\alpha_1} x_1 = \omega^{\beta_1} y_1$, and by the strict right monotonicity of ordinal addition, $cmp(\alpha, \beta) \equiv cmp(\sum_{i=2}^{n} \omega^{\alpha_i} x_i, \sum_{i=2}^{m} \omega^{\beta_i} y_i)$. By the inductive hypothesis, this is equivalent to $\mathbf{cmp_o}(\mathbf{rest}(\mathtt{a}), \mathbf{rest}(\mathtt{b}))$. □

Theorem 13. *For all* $\alpha, \beta \in \epsilon_0$ $\mathrm{CNF}(\alpha + \beta) = \mathrm{CNF}(\alpha) +_o \mathrm{CNF}(\beta)$.

Proof The proof is by induction on α. The intuition of the correctness is based on the definition of additive principal ordinals. Let i be the least index such that $\alpha_i < \beta_1$. Then by the closure of additive principal ordinals under addition, $\sum_{k=i}^{n} \omega^{\alpha_i} x_i < \omega^{\beta_1}$, and thus $\sum_{k=i}^{n} \omega^{\alpha_i} x_i + \omega^{\beta_1} = \omega^{\beta_1}$. So β smashes these terms of α. If $\alpha_{i-1} > \beta_1$, we are done, since we are in CNF. Otherwise, $\alpha_{i-1} = \beta_1$, in which case $\omega^{\alpha_{i-1}} x_{i-1} + \omega^{\beta_1} y1 = \omega^{\alpha_{i-1}}(x_{i-1} + y_1)$ by the left distributive property. Hence, our algorithm recurses down `a` until we find an exponent that's at most the first exponent of `b`. Then we take the appropriate action depending on whether that exponent of `a` is less than or equal to the first exponent of `b`. □

Theorem 14. *For all* $\alpha, \beta \in \epsilon_0$ *such that* $\beta \leq \alpha$, $\mathrm{CNF}(\alpha - \beta) = \mathrm{CNF}(\alpha) -_o \mathrm{CNF}(\beta)$.

Proof The proof requires showing that $\mathrm{CNF}(\beta) +_o \mathrm{CNF}(\alpha) -_o \mathrm{CNF}(\beta) = \mathrm{CNF}(\alpha)$ and $\mathbf{cnfp}(\alpha -_o \beta)$. This follows using induction on $\#\mathrm{CNF}(\alpha)$ and $\#\mathrm{CNF}(\beta)$. □

For the following theorems assume $\mathbf{cnfp}(\mathtt{a})$, $\mathbf{cnfp}(\mathtt{b})$, and $\mathbf{cnfp}(\mathtt{d})$.

Lemma 14. $\mathbf{padd}(\mathtt{a}, \mathtt{b}, \mathbf{c}(\mathtt{a}, \mathtt{b})) = \mathtt{a} +_o \mathtt{b}$

Proof Proof is by induction on $\mathbf{c}(\mathtt{a},\mathtt{b})$. □

Lemma 15. $\mathtt{k} \leq \mathbf{c}(\mathtt{a}, \mathtt{b}) \Rightarrow \mathbf{pmult}(\mathtt{a}, \mathtt{b}, \mathtt{k}) = \mathtt{a} *_o \mathtt{b}$.

Proof This is easily proven by induction on $|\mathtt{b}|$ using Lemma 14. □

Corollary 3. $\mathtt{a} \cdot_o \mathtt{b} = \mathtt{a} *_o \mathtt{b}$.

Lemma 16. *For all* $\alpha, \beta \in \epsilon_0$, $\mathrm{CNF}(\alpha \cdot \beta) = \mathrm{CNF}(\alpha) *_o \mathrm{CNF}(\beta)$.

Proof Clearly, the algorithm is correct when $\alpha = 0$ or $\beta = 0$, and when $\alpha, \beta \in \omega$. Otherwise, we use transfinite induction on β, where the base case has already been established. If $\beta \in \omega$, then $\alpha\beta = \alpha(1 + \beta - 1) = \alpha + \alpha(\beta - 1)$. Using the induction hypothesis and the definitions of $+_o$ and $*_o$, we see that the CNF representation of this ordinal is what $*_o$ returns. Finally, if β is not a natural number, we apply the left distributivity property to distribute α over β: $\alpha\beta = \sum_{i=1}^{m} \alpha\omega^{\beta_i} y_i + \alpha q$. So, since we know how to deal with αq, we only have to deal with terms of the form $\alpha\omega^{\beta_i} y_i$. By the properties of ordinal multiplication and exponentiation, we get for any $z \in \omega$, $\omega^{\alpha_1} z \cdot \omega^{\beta_i} y_i = \omega^{\alpha_1}(z \cdot \omega^{\beta_i}) y_i = \omega^{\alpha_1} \omega^{\beta_i} y_i = \omega^{\alpha_1 + \beta_i} y_i$. Hence, by the left weak monotonicity of multiplication, $\alpha < \omega^{\alpha_1}(x_1 + 1) \quad \Rightarrow \quad \alpha \cdot \omega^{\beta_i} y_i \leq \omega^{\alpha_1 + \beta_i} y_i$ and $\alpha \geq \omega^{\alpha_1} x_1 \quad \Rightarrow \quad \alpha \cdot \omega^{\beta_i} y_i \geq \omega^{\alpha_1 + \beta_i} y_i$. □

Theorem 15. *For all* $\alpha, \beta \in \epsilon_0$, $\mathrm{CNF}(\alpha \cdot \beta) = \mathrm{CNF}(\alpha) \cdot_o \mathrm{CNF}(\beta)$

Proof This follows directly from Corollary 3 and Lemma 16. $\square$

Lemma 17. $\forall p, x \in \omega, \beta \in \epsilon_0$ *such that* $\beta > 0$, $p^{\omega^\beta x} = \omega^{\omega^{\beta-1} x}$

Proof $p^{\omega^\beta x} = p^{\omega^{1+\beta-1} x} = p^{\omega \cdot \omega^{\beta-1} x} = (p^\omega)^{\omega^{\beta-1} x} = \omega^{\omega^{\beta-1} x}$ $\square$

Theorem 16. *For all* $p \in \omega, \beta \in \epsilon_0$ *such that* $\beta \geq \omega$, $\mathrm{CNF}(p^\beta) = \mathbf{exp_1}(\mathrm{CNF}(p), \mathrm{CNF}(\beta))$.

Proof The proof is by induction on β, and uses Lemma 17. $\square$

Lemma 18. *For all* $\mathtt{a}, \mathtt{b}$ *such that* $\mathbf{cnfp}(\mathtt{a})$, $\mathbf{cnfp}(\mathtt{b})$, $\neg(\mathbf{atom}(\mathtt{a}))$ *and* $\mathbf{limitp}(\mathtt{b})$, $\mathtt{a} \cdot_o \mathtt{b} = (\langle \mathbf{fe}(\mathtt{a}), 1\rangle, 0) \cdot_o \mathtt{b}$.

Proof The proof is by induction on $|\mathtt{b}|$. $\square$

Theorem 17. *For all* $\alpha \in \epsilon_0, q \in \omega$ *such that* $\alpha \geq \omega$, $\mathbf{limitp}(\mathrm{CNF}(\alpha))$, *and* $q > 0$, $\mathrm{CNF}(\alpha^q) = \mathbf{exp_2}(\mathrm{CNF}(\alpha), \mathrm{CNF}(q))$.

Proof The proof is by induction on q, using Lemma 18 when $q = 2$. $\square$

Lemma 19. *For all* $\mathtt{a}$ *and* $\mathtt{q} \in \omega$ *such that* $\mathbf{cnfp}(\mathtt{a})$ *and* $\neg(\mathtt{a} <_o \mathrm{CNF}(\omega))$ $\mathbf{helper\text{-}exp_3}(\mathbf{limitpart}(\mathtt{a}), \mathtt{p}, |\mathtt{a}|, \mathtt{q}) = (\sum_{i=0}^{\mathtt{q}-1} \mathbf{exp_2}(\mathbf{limitpart}(\mathtt{a}), \mathtt{q}-i) \cdot_o \mathtt{p}) +_o \mathtt{p}$ *where* $\mathtt{p} = \mathbf{natpart}(\mathtt{a})$ *(and the summation is with respect to* $+_o$*).*

Proof The proof follows fairly easily by induction on $\mathtt{q}$, using Lemma 14. $\square$

Theorem 18. *For all* $\alpha \in \epsilon_0, q \in \omega$ *such that* $\alpha \geq \omega$ *and* $q > 0$, $\mathrm{CNF}(\alpha^q) = \mathbf{exp_3}(\mathrm{CNF}(\alpha), \mathrm{CNF}(q))$.

Proof Let $\delta = \sum_{i=1}^{n} \omega^{\alpha_i} x_i$. Note that $\mathrm{CNF}(\delta) = \mathbf{limitpart}(\mathrm{CNF}(\alpha))$. By Lemma 19, all we need to prove is that $\alpha^q = \delta^q + (\sum_{j=1}^{q-1} \delta^{q-j}) + p$ for all $q > 0$. We will do so by induction on q. If $q = 1$, this is clearly true. Suppose $q > 1$ and $\alpha^{q-1} = \delta^{q-1} + (\sum_{j=1}^{q-2} \delta^{q-1-j}) + p$. Then we get

$$\begin{aligned}
\alpha^q &= \alpha^{q-1} \cdot \alpha \\
&= (\delta^{q-1} + (\textstyle\sum_{j=1}^{q-2} \delta^{q-1-j})p + p) \cdot \alpha \quad \{ \text{ induction hypothesis } \} \\
&= (\delta^{q-1} + (\textstyle\sum_{j=1}^{q-2} \delta^{q-1-j})p + p) \cdot (\delta + p) \\
&= ((\delta^{q-1} + (\textstyle\sum_{j=1}^{q-2} \delta^{q-1-j})p + p) \cdot \delta) + ((\delta^{q-1} + (\textstyle\sum_{j=1}^{q-2} \delta^{q-1-j})p + p) \cdot p)
\end{aligned}$$

Looking at each piece seperately, we let $\gamma = \sum_{j=1}^{q-2} \delta^{q-1-j}$. We then get:

$$\begin{aligned}
&\mathrm{CNF}((\delta^{q-1} + \gamma p + p) \cdot \delta) \\
&= \mathbf{padd}(\mathrm{CNF}(\delta^{q-1}), \mathrm{CNF}((\textstyle\sum_{j=1}^{q-2} \delta^{q-1-j})p + p), n) \cdot_o \mathrm{CNF}(\delta) \\
&= (\langle \mathrm{CNF}(\alpha_1 \cdot (q-1)), 1\rangle, 0) \cdot_o \mathrm{CNF}(\delta) \quad \{ \text{ Lemma 19 } \} \\
&= \mathbf{exp_2}(\mathrm{CNF}(\delta), q) \\
&= \mathrm{CNF}(\delta^q)
\end{aligned}$$

and

$$
\begin{aligned}
&\mathrm{CNF}((\delta^{q-1} + \gamma p + p) \cdot p) \\
&= \mathrm{CNF}((\delta^{q-1} + \gamma p + p)) \cdot_o p \\
&= \mathbf{padd}(\mathrm{CNF}(\delta^{q-1}), \mathrm{CNF}(\gamma p + p), n) \cdot_o p \\
&= \mathbf{padd}(\mathbf{exp_2}(\mathrm{CNF}(\delta), \mathrm{CNF}(q-1)), \mathrm{CNF}(\gamma p + p), n) \cdot_o p \\
&= \mathbf{padd}(\langle \mathrm{CNF}(\alpha_1 \cdot (q-1)), x_1 \rangle \ . \ \mathbf{rest}(\mathbf{exp_2}(\delta, q)), \mathrm{CNF}(\gamma p + p), n) \cdot_o p \\
&= \mathbf{padd}(\langle \mathrm{CNF}(\alpha_1 \cdot (q-1)), x_1 \cdot p \rangle \ . \ \mathbf{rest}(\mathbf{exp_2}(\delta, q)), \mathrm{CNF}(\gamma p + p), n) \\
&= \mathbf{padd}(\langle \mathrm{CNF}(\alpha_1 \cdot (q-1)), x_1 \rangle \ . \ \mathbf{rest}(\mathbf{exp_2}(\delta, q)) \cdot_o p, \mathrm{CNF}(\gamma p + p), n) \\
&= \mathbf{padd}(\mathrm{CNF}(\delta^{q-1} p), \mathrm{CNF}(\gamma p + p), n) \\
&= \mathrm{CNF}((\textstyle\sum_{j=0}^{q-2} \delta^{q-1-j}) p + p) \\
&= \mathrm{CNF}((\textstyle\sum_{j=1}^{q-1} \delta^{q-j}) p + p)
\end{aligned}
$$

Hence, we have $\alpha^q = \delta^q + (\sum_{j=1}^{q-1} \delta^{q-j})p + p$. $\square$

Theorem 19. *For all $\alpha, \beta \in \epsilon_0$, such that $\alpha, \beta \geq \omega$, $\mathrm{CNF}(\alpha^\beta) = \mathbf{exp_4}(\mathrm{CNF}(\alpha), \mathrm{CNF}(\beta))$.*

Proof $\mathrm{CNF}(\alpha^\beta) = \mathrm{CNF}(\alpha^{\sum_{i=1}^m \omega^{\beta_i} y_i + q}) = \mathrm{CNF}(\prod_{i=1}^m \alpha^{\omega^{\beta_i} y_i} \cdot \alpha^q)$. Now given any $\xi \in \epsilon_0$ and $z \in \omega$, consider the following two inequalities:

$$
\begin{aligned}
&\alpha < \omega^{\alpha_1+1} \Rightarrow \alpha^{\omega^\xi z} \leq (\omega^{\alpha_1+1})^{\omega^\xi z} \Rightarrow \alpha^{\omega^\xi z} \leq \omega^{(\alpha_1+1)\cdot\omega^\xi z} \Rightarrow \alpha^{\omega^\xi z} \leq \omega^{\alpha_1 \cdot \omega^\xi z} \\
&\alpha > \omega^{\alpha_1} \Rightarrow \alpha^{\omega^\xi z} \geq (\omega^{\alpha_1})^{\omega^\xi z} \Rightarrow \alpha^{\omega^\xi z} \geq \omega^{(\alpha_1)\cdot\omega^\xi z} \Rightarrow \alpha^{\omega^\xi z} \geq \omega^{\alpha_1 \cdot \omega^\xi z}
\end{aligned}
$$

Hence,

$$
\begin{aligned}
\mathrm{CNF}(\alpha^\beta) &= \mathrm{CNF}(\textstyle\prod_{i=1}^m \alpha^{\omega^{\beta_i} y_i} \cdot \alpha^q) \\
&= \mathrm{CNF}(\textstyle\prod_{i=1}^m \omega^{\alpha_1 \cdot \omega^{\beta_i} y_i} \cdot \alpha^q) \\
&= \mathrm{CNF}(\omega^{\alpha_1 \cdot \sum_{i=1}^m \omega^{\beta_i} y_i} \cdot \alpha^q) \\
&= (\langle \mathbf{fe}(\mathrm{CNF}(\alpha)) \cdot_o \mathbf{limitpart}(\mathrm{CNF}(\beta)), 1 \rangle, 0) \cdot_o \mathbf{exp_2}(\alpha, q) \\
&= \mathbf{exp_4}(\mathrm{CNF}(\alpha), \mathrm{CNF}(\beta)) \quad \square
\end{aligned}
$$

Theorem 20. *For all $\alpha, \beta \in \epsilon_0, \mathrm{CNF}(\alpha^\beta) = \mathbf{exp_o}(\mathrm{CNF}(\alpha), \mathrm{CNF}(\beta))$.*

Proof The proof follows from Theorems 16, 18, and 19. $\square$

5 Conclusion

We presented efficient algorithms for ordinal addition, subtraction, multiplication, and exponentiation on succinct ordinal representations, proved their correctness, and analyzed their complexity. We implemented a version of the algorithms in the ACL2 system, mechanically verified their correctness, and developed a library of theorems that can be used to significantly automate reasoning involving the ordinals [22]. This library is part of the current ACL2 distribution [16]. While the theory of the ordinal numbers has been studied by various research communities for over 100 years, we believe that we are the first to provide a complete solution to the ordinal arithmetic problem, *i.e.*, to give algorithms for ordinal arithmetic on ordinal notations.

Our work can be extended by replacing ϵ_0 with larger countable ordinals. No ordinal notation can represent all countable ordinals but there are well known notations that can represent ordinals up to Γ_0 (which is needed to show termination of some term rewrite systems [9, 12]) and further into the Veblen hierarchies [35] and further still [24, 30, 31]. Another possible extension is to define additional operations on ordinals, *e.g.*, division, taking logs, etc.

References

1. F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
2. A. Beckmann, S. R. Buss, and C. Pollett. Ordinal notations and well-orderings in bounded arithmetic. *Annals of Pure and Applied Logic*, pages 197–223, 2003.
3. B. Brock, M. Kaufmann, and J. S. Moore. ACL2 theorems about commercial microprocessors. In M. Srivas and A. Camilleri, editors, *Formal Methods in Computer-Aided Design (FMCAD'96)*, pages 275–293. Springer-Verlag, 1996.
4. G. Cantor. Beiträge zur Begründung der transfiniten Mengenlehre. *Mathematische Annalen*, xlvi:481–512, 1895.
5. G. Cantor. Beiträge zur Bgründung der transfiniten Mengenlehre. *Mathematische Annalen*, xlix:207–246, 1897.
6. G. Cantor. *Contributions to the Founding of the Theory of Transfinite Numbers*. Dover Publications, Inc., 1952. Translated by Philip E. B. Jourdain.
7. A. Church and S. C. Kleene. Formal definitions in the theory of ordinal numbers. *Fundamenta mathematicae*, 28:11–21, 1937.
8. N. Dershowitz. Trees, ordinals, and termination. In *Proceedings of the 4th International Joint Conference on Theory and Practice of Software Development (TAPSOFT)*, volume 668 of *LNCS*, pages 243–250. Springer-Verlag, April 1993.
9. N. Dershowitz and M. Okada. Proof-theoritic techniques for term rewriting theory. In *3rd IEEE Symp. on Logic in Computer Science*, pages 104–111, 1988.
10. N. Dershowitz and E. M. Reingold. Ordinal arithmetic with list structures. In *Logical Foundations of Computer Science*, pages 117–126, 1992.
11. K. Devlin. *The Joy of Sets: Fundamentals of Contemporary Set Theory*. Springer-Verlag, second edition, 1992.
12. J. H. Gallier. What's so special about Kruskal's theorem and the ordinal Γ_0? A survey of some results in proof theory. *Annals of Pure and Applied Logic*, pages 199–260, 1991.
13. G. Gentzen. Die widerspruchsfreiheit der reinen zahlentheorie. *Mathematische Annalen*, 112:493–565, 1936. English translation in M. E. Szabo (ed.), The Collected Works of Gerhard Gentzen, pp. 132-213, North Holland, Amsterdam, 1969.
14. D. A. Greve. Symbolic simulation of the JEM1 microprocessor. In *Formal Methods in Computer-Aided Design – FMCAD*, LNCS. Springer-Verlag, 1998.
15. M. Kaufmann, P. Manolios, and J. S. Moore. *Computer-Aided Reasoning: An Approach*. Kluwer Academic Publishers, July 2000.
16. M. Kaufmann and J. S. Moore. ACL2 homepage. See URL `http://www.cs.-utexas.edu/users/moore/acl2`.

17. M. Kaufmann and J. S. Moore, editors. *Proceedings of the ACL2 Workshop 2000*. The University of Texas at Austin, Technical Report TR-00-29, November 2000.
18. K. Kunen. *Set Theory - an Introduction to Independence Proofs*, volume 102 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 1980.
19. P. Manolios. Correctness of pipelined machines. In W. A. Hunt, Jr. and S. D. Johnson, editors, *Formal Methods in Computer-Aided Design–FMCAD 2000*, volume 1954 of *LNCS*, pages 161–178. Springer-Verlag, 2000.
20. P. Manolios. *Mechanical Verification of Reactive Systems*. PhD thesis, University of Texas at Austin, August 2001.
21. P. Manolios, K. Namjoshi, and R. Sumners. Linking theorem proving and model-checking with well-founded bisimulation. In N. Halbwachs and D. Peled, editors, *Computer-Aided Verification–CAV '99*, volume 1633 of *LNCS*, pages 369–379. Springer-Verlag, 1999.
22. P. Manolios and D. Vroon. Ordinal arithmetic in ACL2, 2003. Submitted to the 4th International Workshop on the ACL2 Theorem Prover and Its Applications.
23. I. Medina-Bulo, F. Palomo-Lozano, and J. A. Alonso-Jimenez. Implementation in ACL2 of well-founded polynomial orderings. In M. Kaufmann and J. S. Moore, editors, *Proceedings of the ACL2 Workshop 2002*. 2002.
24. L. W. Miller. Normal functions and constructive ordinal notations. *Journal of Symbolic Logic*, 41:439–459, June 1976.
25. J. S. Moore, T. Lynch, and M. Kaufmann. A mechanically checked proof of the AMD5_K86 floating-point division program. *IEEE Trans. Comp.*, 47(9):913–926, September 1998.
26. H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. MIT Press, 1st paperback edition, 1987.
27. J.-L. Ruiz-Reina, J.-A. Alonso, M.-J. Hidalgo, and F.-J. Martin. Multiset relations: A tool for proving termination. In Kaufmann and Moore [17].
28. D. M. Russinoff. A mechanically checked proof of IEEE compliance of a register-transfer-level specification of the AMD-K7 floating-point multiplication, division, and square root instructions. *London Mathematical Society Journal of Computation and Mathematics*, 1:148–200, December 1998.
29. K. Schütte. *Proof Theory*. Springer-Verlag, 1977. translated by J. N. Crossley from the revised version of *Beweistheorie*, 1st edition, 1960.
30. A. Setzer. Ordinal systems. In B. Cooper and J. Truss, editors, *Sets and Proofs*, pages 301–331. Cambridge University Press, 1999.
31. A. Setzer. Ordinal systems part 2: One inaccessible. In *Logic Colloquium '98*, volume 13 of *ASL Lecture Notes in Logic*, pages 426–448, 2000.
32. R. Sumners. An incremental stuttering refinement proof of a concurrent program in ACL2. In Kaufmann and Moore [17].
33. M. Sustik. Proof of Dixon's lemma using the ACL2 theorem prover via an explicit ordinal mapping, 2003. Submitted to the 4th International Workshop on the ACL2 Theorem Prover and Its Applications.
34. A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*. Cambridge University Press, second edition, 2000.
35. O. Veblen. Continuous increasing functions of finite and transfinite ordinals. *Transactions of the American Mathematical Society*, 9:280–292, 1908.