

Positional Notation for Natural Numbers in an Arbitrary Base

Charles Staats III

May 14, 2024

Abstract

We demonstrate the existence and uniqueness of the base- n representation of a natural number, where n is any natural number greater than 1. This comes up when trying to translate mathematical contest problems and solutions into Isabelle/HOL.

Contents

1	Infinite sums	1
2	Modular arithmetic	3
3	Digits as sequence	4
4	Little Endian notation	8
5	Big Endian notation	9
6	Exercises	11

theory *DigitsInBase*

imports *HOL-Computational-Algebra.Computational-Algebra HOL-Number-Theory.Number-Theory*

begin

1 Infinite sums

In this section, it is shown that an infinite series *of natural numbers* converges if and only if its terms are eventually zero. Additionally, the notion of a summation starting from an index other than zero is defined. A few obvious lemmas about these notions are established.

definition *eventually-zero* :: (nat \Rightarrow -) \Rightarrow bool **where**
eventually-zero (D :: nat \Rightarrow -) \longleftrightarrow ($\forall_{\infty} n. D n = 0$)

lemma *eventually-zero-char*:

shows *eventually-zero* $D \longleftrightarrow (\exists s. \forall i \geq s. D\ i = 0)$
 ⟨*proof*⟩

There's a lot of commonality between this setup and univariate polynomials, but drawing out the similarities and proving them is beyond the scope of the current version of this theory except for the following lemma.

lemma *eventually-zero-poly*:
shows *eventually-zero* $D \longleftrightarrow D = \text{poly.coeff } (Abs\text{-poly } D)$
 ⟨*proof*⟩

lemma *eventually-zero-imp-summable* [*simp*]:
assumes *eventually-zero* D
shows *summable* D
 ⟨*proof*⟩

lemma *summable-bounded*:
fixes *my-seq* :: $\text{nat} \Rightarrow \text{nat}$ **and** $n :: \text{nat}$
assumes $\bigwedge i. i \geq n \longrightarrow \text{my-seq } i = 0$
shows *summable* *my-seq*
 ⟨*proof*⟩

lemma *sum-bounded*:
fixes *my-seq* :: $\text{nat} \Rightarrow \text{nat}$ **and** $n :: \text{nat}$
assumes $\bigwedge i. i \geq n \longrightarrow \text{my-seq } i = 0$
shows $(\sum i. \text{my-seq } i) = (\sum i < n. \text{my-seq } i)$
 ⟨*proof*⟩

lemma *product-seq-eventually-zero*:
fixes *seq1 seq2* :: $\text{nat} \Rightarrow \text{nat}$
assumes *eventually-zero* *seq1*
shows *eventually-zero* $(\lambda i. \text{seq1 } i * \text{seq2 } i)$
 ⟨*proof*⟩

abbreviation *upper-sum*
where *upper-sum* $\text{seq } n \equiv \sum i. \text{seq } (i + n)$
syntax
 -*from-sum* :: $\text{idt} \Rightarrow 'a \Rightarrow 'b \Rightarrow 'b$ $((\exists \sum \text{-}\geq \text{-} / \text{-}) [0,0,10] 10)$
translations
 $\sum i \geq n. t == \text{CONST upper-sum } (\lambda i. t) n$

The following two statements are proved as a sanity check. They are not intended to be used anywhere.

corollary
fixes *seq* :: $\text{nat} \Rightarrow \text{nat}$ **and** $a :: \text{nat}$
assumes *seq-def*: $\bigwedge i. \text{seq } i = (\text{if } i = 0 \text{ then } a \text{ else } 0)$
shows $(\sum i \geq 0. \text{seq } i) = \text{upper-sum } (\lambda i. \text{seq } i) 0$
 ⟨*proof*⟩

corollary

fixes $seq :: nat \Rightarrow nat$ **and** $a :: nat$
assumes $seq-def: \bigwedge i. seq\ i = (if\ i = 0\ then\ a\ else\ 0)$
shows $(\sum_{i \geq 0}. seq\ i) = a$
 $\langle proof \rangle$

lemma *bounded-sum-from*:
fixes $seq :: nat \Rightarrow nat$ **and** $n\ s :: nat$
assumes $\forall i > s. seq\ i = 0$ **and** $n \leq s$
shows $(\sum_{i \geq n}. seq\ i) = (\sum_{i=n..s}. seq\ i)$
 $\langle proof \rangle$

lemma *split-suminf*:
fixes $seq :: nat \Rightarrow nat$ **and** $n :: nat$
assumes *eventually-zero* seq
shows $(\sum i. seq\ i) = (\sum_{i < n}. seq\ i) + (\sum_{i \geq n}. seq\ i)$
 $\langle proof \rangle$

lemma *dvd-suminf*:
fixes $seq :: nat \Rightarrow nat$ **and** $b :: nat$
assumes *eventually-zero* seq **and** $\bigwedge i. b\ dvd\ seq\ i$
shows $b\ dvd\ (\sum i. seq\ i)$
 $\langle proof \rangle$

lemma *eventually-zero-shifted*:
assumes *eventually-zero* seq
shows *eventually-zero* $(\lambda i. seq\ (i + n))$
 $\langle proof \rangle$

2 Modular arithmetic

This section establishes a number of lemmas about modular arithmetic, including that modular division distributes over an “infinite” sum whose terms are eventually zero.

lemma *pmod-int-char*:
fixes $a\ b\ d :: int$
shows $[a = b] \ (mod\ d) \longleftrightarrow (\exists (n::int). a = b + n*d)$
 $\langle proof \rangle$

lemma *equiv-conj-if*:
assumes $P \Longrightarrow Q$ **and** $P \Longrightarrow R$ **and** $Q \Longrightarrow R \Longrightarrow P$
shows $P \longleftrightarrow Q \wedge R$
 $\langle proof \rangle$

lemma *mod-successor-char*:
fixes $k\ k'\ i\ b :: nat$
assumes $(b::nat) \geq 2$
shows $[k = k'] \ (mod\ b^{Suc\ i}) \longleftrightarrow [k\ div\ b^i = k'\ div\ b^i] \ (mod\ b) \wedge [k = k'] \ (mod\ b^i)$

<proof>

lemma *mod-1*:

fixes $k\ k'\ b :: \text{nat}$

shows $[k = k'] \pmod{b \wedge 0}$

<proof>

lemma *mod-distributes*:

fixes $\text{seq} :: \text{nat} \Rightarrow \text{nat}$ **and** $b :: \text{nat}$

assumes $\exists n. \forall i \geq n. \text{seq } i = 0$

shows $[(\sum i. \text{seq } i) = (\sum i. \text{seq } i \pmod{b})] \pmod{b}$

<proof>

lemma *another-mod-cancellation-int*:

fixes $a\ b\ c\ d\ m :: \text{int}$

assumes $d > 0$ **and** $[m = a + b] \pmod{c * d}$ **and** $a \text{ div } d = 0$ **and** $d \text{ dvd } b$

shows $[m \text{ div } d = b \text{ div } d] \pmod{c}$

<proof>

lemma *another-mod-cancellation*:

fixes $a\ b\ c\ d\ m :: \text{nat}$

assumes $d > 0$ **and** $[m = a + b] \pmod{c * d}$ **and** $a \text{ div } d = 0$ **and** $d \text{ dvd } b$

shows $[m \text{ div } d = b \text{ div } d] \pmod{c}$

<proof>

3 Digits as sequence

Rules are introduced for computing the i^{th} digit of a base- b representation and the number of digits required to represent a given number. (The latter is essentially an integer version of the base- b logarithm.) It is shown that the sum of the terms $d_i b^i$ converges to m if d_i is the i^{th} digit m . It is shown that the sequence of digits defined is the unique sequence of digits less than b with this property

Additionally, the `digits_in_base` locale is introduced, which specifies a single symbol b referring to a natural number greater than one (the base of the digits). Consequently this symbol is omitted from many of the following lemmas and definitions.

locale *digits-in-base* =

fixes $b :: \text{nat}$

assumes *b-gte-2*: $b \geq 2$

begin

lemma *b-facts* [*simp*]:

shows $b > 1$ **and** $b > 0$ **and** $b \neq 1$ **and** $b \neq 0$ **and** $1 \pmod{b} = 1$ **and** $1 \text{ div } b = 0$

<proof>

Definition based on [1].

abbreviation $ith_digit :: nat \Rightarrow nat \Rightarrow nat$ **where**
 $ith_digit\ m\ i \equiv (m\ div\ b^i) \bmod\ b$

lemma $ith_digit_lt_base$:
fixes $m\ i :: nat$
shows $0 \leq ith_digit\ m\ i$ **and** $ith_digit\ m\ i < b$
 $\langle proof \rangle$

definition $num_digits :: nat \Rightarrow nat$
where $num_digits\ m = (LEAST\ i.\ m < b^i)$

lemma num_digits_works :
shows $m < b^{(num_digits\ m)}$
 $\langle proof \rangle$

lemma num_digits_le :
assumes $m < b^i$
shows $num_digits\ m \leq i$
 $\langle proof \rangle$

lemma num_digits_zero :
fixes $m :: nat$
assumes $num_digits\ m = 0$
shows $m = 0$
 $\langle proof \rangle$

lemma num_digits_gt :
assumes $m \geq b^i$
shows $num_digits\ m > i$
 $\langle proof \rangle$

lemma num_digits_eqI [*intro*]:
assumes $m \geq b^i$ **and** $m < b^{(i+1)}$
shows $num_digits\ m = i + 1$
 $\langle proof \rangle$

lemma num_digits_char :
assumes $m \geq 1$
shows $num_digits\ m = i + 1 \iff m \geq b^i \wedge m < b^{(i+1)}$
 $\langle proof \rangle$

Statement based on [1].

lemma $num_digits_recurrence$:
fixes $m :: nat$
assumes $m \geq 1$
shows $num_digits\ m = num_digits\ (m\ div\ b) + 1$
 $\langle proof \rangle$

lemma num_digits_zero-2 [*simp*]: $num_digits\ 0 = 0$

```

    <proof>

end

locale base-10
begin

    As a sanity check, the number of digits in base ten is computed for several
    natural numbers.

sublocale digits-in-base 10
    <proof>

corollary
    shows num-digits 0 = 0
        and num-digits 1 = 1
        and num-digits 9 = 1
        and num-digits 10 = 2
        and num-digits 99 = 2
        and num-digits 100 = 3
    <proof>

end

context digits-in-base
begin

lemma high-digits-zero-helper:
    fixes m i :: nat
    shows i < num-digits m  $\vee$  ith-digit m i = 0
    <proof>

lemma high-digits-zero:
    fixes m i :: nat
    assumes i  $\geq$  num-digits m
    shows ith-digit m i = 0
    <proof>

lemma digit-expansion-bound:
    fixes i :: nat and A :: nat  $\Rightarrow$  nat
    assumes  $\bigwedge j. A j < b$ 
    shows  $(\sum j < i. A j * b^j) < b^i$ 
    <proof>

    Statement and proof based on [1].

lemma num-digits-suc:
    fixes n m :: nat
    assumes Suc n = num-digits m
    shows n = num-digits (m div b)
    <proof>

```

Proof (and to some extent statement) based on [1].

lemma *digit-expansion-bounded-seq*:

fixes $m :: nat$

shows $m = (\sum_{j < num-digits\ m} ith-digit\ m\ j * b^{j})$

<proof>

A natural number can be obtained from knowing all its base- b digits by the formula $\sum_j d_j b^j$.

theorem *digit-expansion-seq*:

fixes $m :: nat$

shows $m = (\sum_j. ith-digit\ m\ j * b^{j})$

<proof>

lemma *lower-terms*:

fixes $a\ c\ i :: nat$

assumes $c < b^{i}$ **and** $a < b$

shows $ith-digit\ (a * b^{i} + c)\ i = a$

<proof>

lemma *upper-terms*:

fixes $A\ a\ i :: nat$

assumes $b * b^{i} \text{ dvd } A$ **and** $a < b$

shows $ith-digit\ (A + a * b^{i})\ i = a$

<proof>

lemma *current-term*:

fixes $A\ a\ c\ i :: nat$

assumes $b * b^{i} \text{ dvd } A$ **and** $c < b^{i}$ **and** $a < b$

shows $ith-digit\ (A + a * b^{i} + c)\ i = a$

<proof>

Given that

$$m = \sum_i d_i b^i$$

where the d_i are all natural numbers less than b , it follows that d_j is the j^{th} digit of m .

theorem *seq-uniqueness*:

fixes $m\ j :: nat$ **and** $D :: nat \Rightarrow nat$

assumes *eventually-zero* D **and** $m = (\sum_i. D\ i * b^{i})$ **and** $\bigwedge i. D\ i < b$

shows $D\ j = ith-digit\ m\ j$

<proof>

end

4 Little Endian notation

In this section we begin to define finite digit expansions. Ultimately we want to write digit expansions in “big endian” notation, by which we mean with the highest place digit on the left and the ones digit on the right, since this ordering is standard in informal mathematics. However, it is easier to first define “little endian” expansions with the ones digit on the left since that way the list indexing agrees with the sequence indexing used in the previous section (whenever both are defined).

Notation, definitions, and lemmas in this section typically start with the prefix **LE** (for “little endian”) to distinguish them from the big endian versions in the next section.

fun *LEeval-as-base* (-LEbase - [65, 65] 70)
where [] LEbase b = 0
 | (d # d-list) LEbase b = d + b * (d-list LEbase b)

corollary shows [2, 4] LEbase 5 = (22::nat)
 ⟨proof⟩

lemma *LEbase-one-digit* [simp]: **shows** [a::nat] LEbase b = a
 ⟨proof⟩

lemma *LEbase-two-digits* [simp]: **shows** [a₀::nat, a₁] LEbase b = a₀ + a₁ * b
 ⟨proof⟩

lemma *LEbase-three-digits* [simp]: **shows** [a₀::nat, a₁, a₂] LEbase b = a₀ + a₁*b + a₂*b²
 ⟨proof⟩

lemma *LEbase-closed-form*:
shows (A :: nat list) LEbase b = (∑ i < length A . A[i] * bⁱ)
 ⟨proof⟩

lemma *LEbase-concatenate*:
fixes A D :: nat list **and** b :: nat
shows (A @ D) LEbase b = (A LEbase b) + b^(length A) * (D LEbase b)
 ⟨proof⟩

context *digits-in-base*
begin

definition *LEdigits* :: nat ⇒ nat list **where**
 LEdigits m = [ith-digit m i . i ← [0..<(num-digits m)]]

lemma *length-is-num-digits*:
fixes m :: nat
shows length (LEdigits m) = num-digits m

<proof>

lemma *ith-list-element* [simp]:
 assumes $(i::nat) < length (LEdigits\ m)$
 shows $(LEdigits\ m)\ !\ i = ith-digit\ m\ i$
 <proof>

lemma *LEbase-infinite-sum*:
 fixes $m :: nat$
 shows $(\sum\ i.\ ith-digit\ m\ i * b^i) = (LEdigits\ m)_{LEbase\ b}$
 <proof>

lemma *digit-expansion-LElist*:
 fixes $m :: nat$
 shows $(LEdigits\ m)_{LEbase\ b} = m$
 <proof>

lemma *LElist-uniqueness*:
 fixes $D :: nat\ list$
 assumes $\forall\ i < length\ D.\ D!\ i < b$ **and** $D = [] \vee last\ D \neq 0$
 shows $LEdigits\ (D_{LEbase\ b}) = D$
 <proof>

lemma *LE-digits-zero* [simp]: $LEdigits\ 0 = []$
 <proof>

lemma *LE-units-digit* [simp]:
 assumes $(m::nat) \in \{1..<b\}$
 shows $LEdigits\ m = [m]$
 <proof>

end

5 Big Endian notation

In this section the desired representation of natural numbers, as finite lists of digits with the highest place on the left, is finally realized.

definition *BEeval-as-base* ($-_{base\ b}$ [65, 65] 70)
 where [simp]: $D_{base\ b} = (rev\ D)_{LEbase\ b}$

corollary **shows** $[4, 2]_{base\ 5} = (22::nat)$
 <proof>

lemma *BEbase-one-digit* [simp]: **shows** $[a::nat]_{base\ b} = a$
 <proof>

lemma *BEbase-two-digits* [simp]: **shows** $[a_1::nat, a_0]_{base\ b} = a_1*b + a_0$
 <proof>

lemma *BEbase-three-digits* [*simp*]: **shows** $[a_2::nat, a_1, a_0]_{base\ b} = a_2 * b^2 + a_1 * b + a_0$
 <proof>

lemma *BEbase-closed-form*:
fixes $A :: nat\ list$ **and** $b :: nat$
shows $A_{base\ b} = (\sum i < length\ A. A!i * b^{(length\ A - Suc\ i)})$
 <proof>

lemma *BEbase-concatenate*:
fixes $A\ D :: nat\ list$ **and** $b :: nat$
shows $(A @ D)_{base\ b} = (A_{base\ b}) * b^{(length\ D)} + (D_{base\ b})$
 <proof>

context *digits-in-base*
begin

definition *digits* :: $nat \Rightarrow nat\ list$ **where**
 $digits\ m = rev\ (LEdigits\ m)$

lemma *length-is-num-digits-2*:
fixes $m :: nat$
shows $length\ (digits\ m) = num-digits\ m$
 <proof>

lemma *LE-BE-equivalence*:
fixes $m :: nat$
shows $(digits\ m)_{base\ b} = (LEdigits\ m)_{LEbase\ b}$
 <proof>

lemma *BEbase-infinite-sum*:
fixes $m :: nat$
shows $(\sum i. ith-digit\ m\ i * b^i) = (digits\ m)_{base\ b}$
 <proof>

Every natural number can be represented in base b , specifically by the digits sequence defined earlier.

theorem *digit-expansion-list*:
fixes $m :: nat$
shows $(digits\ m)_{base\ b} = m$
 <proof>

If two natural numbers have the same base- b representation, then they are equal.

lemma *digits-cancellation*:
fixes $k\ m :: nat$
assumes $digits\ k = digits\ m$
shows $k = m$

<proof>

Suppose we have a finite (possibly empty) sequence D_1, \dots, D_n of natural numbers such that $0 \leq D_i < b$ for all i and such that D_1 , if it exists, is nonzero. Then this sequence is the base- b representation for $\sum_i D_i b^{n-i}$.

theorem *list-uniqueness*:

fixes $D :: \text{nat list}$

assumes $\forall d \in \text{set } D. d < b$ **and** $D = [] \vee D!0 \neq 0$

shows $\text{digits } (D_{\text{base } b}) = D$

<proof>

We now prove some simplification rules (including a recurrence relation) to make it easier for Isabelle/HOL to compute the base- b representation of a natural number.

The base- b representation of 0 is empty, at least following the conventions of this theory file.

lemma *digits-zero [simp]*:

shows $\text{digits } 0 = []$

<proof>

If $0 < m < b$, then the base- b representation of m consists of a single digit, namely m itself.

lemma *single-digit-number [simp]*:

assumes $m \in \{0 < .. < b\}$

shows $\text{digits } m = [m]$

<proof>

For all $m \geq b$, the base- b representation of m consists of the base- b representation of $\lfloor m/b \rfloor$ followed by (as the last digit) the remainder of m when divided by b .

lemma *digits-recurrence [simp]*:

assumes $m \geq b$

shows $\text{digits } m = (\text{digits } (m \text{ div } b)) @ [m \text{ mod } b]$

<proof>

end

6 Exercises

This section contains demonstrations of how to denote certain facts with the notation of the previous sections, and how to quickly prove those facts using the lemmas and theorems above.

The base-5 representation of 22 is 42_5 .

corollary *digits-in-base*. $\text{digits } 5 \ 22 = [4, 2]$

<proof>

A different proof of the same statement.

corollary *digits-in-base.digits* 5 22 = [4, 2]
<proof>

end

References

- [1] B. Porter. Threedivides. <https://isabelle.in.tum.de/dist/library/HOL/HOL-ex/ThreeDivides.html>, 2005. Accessed: 2023-03-06.