Positional Notation for Natural Numbers in an Arbitrary Base

Charles Staats III

May 26, 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

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* ⇒ -) ⇒ *bool* **where**
eventually-zero (*D* :: *nat* ⇒ -) ⟷ (∀$_\infty$ *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* $\longleftrightarrow$ $D = poly.coeff\ (Abs\text{-}poly\ D)$
⟨*proof*⟩

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

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

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

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

abbreviation *upper-sum*
where $upper\text{-}sum\ seq\ n \equiv \sum i.\ seq\ (i + n)$
syntax
-from-sum :: *idt* ⇒ *'a* ⇒ *'b* ⇒ *'b* $((3\sum\text{-}{\geq}\text{-}./\ \text{-})\ [0,0,10]\ 10)$
translations
$\sum i{\geq}n.\ t == CONST\ upper\text{-}sum\ (\lambda i.\ t)\ n$

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

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

corollary

fixes *seq* :: *nat* ⇒ *nat* **and** *a* :: *nat*
assumes *seq-def*: ⋀ *i*. *seq i* = (*if i* = *0 then a else 0*)
shows (∑ *i*≥*0*. *seq i*) = *a*
⟨*proof*⟩

lemma *bounded-sum-from*:
fixes *seq* :: *nat* ⇒ *nat* **and** *n s* :: *nat*
assumes ∀ *i*>*s*. *seq i* = *0* **and** *n* ≤ *s*
shows (∑ *i*≥*n*. *seq i*) = (∑ *i*=*n*..*s*. *seq i*)
⟨*proof*⟩

lemma *split-suminf*:
fixes *seq* :: *nat* ⇒ *nat* **and** *n* :: *nat*
assumes *eventually-zero seq*
shows (∑ *i*. *seq i*) = (∑ *i*<*n*. *seq i*) + (∑ *i*≥*n*. *seq i*)
⟨*proof*⟩

lemma *dvd-suminf*:
fixes *seq* :: *nat* ⇒ *nat* **and** *b* :: *nat*
assumes *eventually-zero seq* **and** ⋀*i*. *b dvd seq i*
shows *b dvd* (∑ *i*. *seq i*)
⟨*proof*⟩

lemma *eventually-zero-shifted*:
assumes *eventually-zero seq*
shows *eventually-zero* (λ *i*. *seq* (*i* + *n*))
⟨*proof*⟩

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*) ⟷ (∃(*n*::*int*). *a* = *b* + *n*∗*d*)
⟨*proof*⟩

lemma *equiv-conj-if*:
assumes *P* ⟹ *Q* **and** *P* ⟹ *R* **and** *Q* ⟹ *R* ⟹ *P*
shows *P* ⟷ *Q* ∧ *R*
⟨*proof*⟩

lemma *mod-successor-char*:
fixes *k k′ i b* :: *nat*
assumes (*b*::*nat*) ≥ *2*
shows [*k* = *k′*] (*mod b*^(*Suc i*)) ⟷ [*k div b*^*i* = *k′ div b*^*i*] (*mod b*) ∧ [*k* = *k′*] (*mod b*^*i*)

⟨*proof*⟩

lemma *mod-1*:
fixes *k k′ b* :: *nat*
shows [*k* = *k′*] (*mod b*^*0*)
⟨*proof*⟩

lemma *mod-distributes*:
fixes *seq* :: *nat* ⇒ *nat* **and** *b* :: *nat*
assumes ∃ *n*. ∀ *i*≥*n*. *seq i* = *0*
shows [(∑ *i*. *seq i*) = (∑ *i*. *seq i mod b*)] (*mod b*)
⟨*proof*⟩

lemma *another-mod-cancellation-int*:
fixes *a b c d m* :: *int*
assumes *d* > *0* **and** [*m* = *a* + *b*] (*mod c* ∗ *d*) **and** *a div d* = *0* **and** *d dvd b*
shows [*m div d* = *b div d*] (*mod c*)
⟨*proof*⟩

lemma *another-mod-cancellation*:
fixes *a b c d m* :: *nat*
assumes *d* > *0* **and** [*m* = *a* + *b*] (*mod c* ∗ *d*) **and** *a div d* = *0* **and** *d dvd b*
shows [*m div d* = *b div d*] (*mod 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* :: *nat*
assumes *b-gte-2*: *b* ≥ *2*
begin

lemma *b-facts* [*simp*]:
shows *b* > *1* **and** *b* > *0* **and** *b* ≠ *1* **and** *b* ≠ *0* **and** *1 mod b* = *1* **and** *1 div b* = *0*
⟨*proof*⟩

Definition based on [1].

abbreviation *ith-digit* :: *nat* ⇒ *nat* ⇒ *nat* **where**
ith-digit m i ≡ (*m div b*^*i*) *mod b*

lemma *ith-digit-lt-base*:
fixes *m i* :: *nat*
shows *0* ≤ *ith-digit m i* **and** *ith-digit m i* < *b*
⟨*proof*⟩

definition *num-digits* :: *nat* ⇒ *nat*
where *num-digits m* = (*LEAST i*. *m* < *b*^*i*)

lemma *num-digits-works*:
shows *m* < *b* ^ (*num-digits m*)
⟨*proof*⟩

lemma *num-digits-le*:
assumes *m* < *b*^*i*
shows *num-digits m* ≤ *i*
⟨*proof*⟩

lemma *num-digits-zero*:
fixes *m* :: *nat*
assumes *num-digits m* = *0*
shows *m* = *0*
⟨*proof*⟩

lemma *num-digits-gt*:
assumes *m* ≥ *b*^*i*
shows *num-digits m* > *i*
⟨*proof*⟩

lemma *num-digits-eqI* [*intro*]:
assumes *m* ≥ *b*^*i* **and** *m* < *b*^(*i*+*1*)
shows *num-digits m* = *i* + *1*
⟨*proof*⟩

lemma *num-digits-char*:
assumes *m* ≥ *1*
shows *num-digits m* = *i* + *1* ⟷ *m* ≥ *b*^*i* ∧ *m* < *b*^(*i*+*1*)
⟨*proof*⟩

Statement based on [1].

lemma *num-digits-recurrence*:
fixes *m* :: *nat*
assumes *m* ≥ *1*
shows *num-digits m* = *num-digits* (*m div b*) + *1*
⟨*proof*⟩

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* ∨ *ith-digit m i* = *0*
⟨*proof*⟩

lemma *high-digits-zero*:
fixes *m i* :: *nat*
assumes *i* ≥ *num-digits m*
shows *ith-digit m i* = *0*
⟨*proof*⟩

lemma *digit-expansion-bound*:
fixes *i* :: *nat* **and** *A* :: *nat* ⇒ *nat*
assumes ⋀*j*. *A j* < *b*
shows (∑*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 $m = (\sum j{<}\textit{num-digits}\ m.\ \textit{ith-digit}\ m\ j * b\hat{}j)$
⟨*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 $m = (\sum j.\ \textit{ith-digit}\ m\ j * b\hat{}j)$
⟨*proof*⟩

lemma *lower-terms*:
fixes $a\ c\ i$ :: *nat*
assumes $c < b\hat{}i$ **and** $a < b$
shows $\textit{ith-digit}\ (a * b\hat{}i + c)\ i = a$
⟨*proof*⟩

lemma *upper-terms*:
fixes $A\ a\ i$ :: *nat*
assumes $b{*}b\hat{}i\ \textit{dvd}\ A$ **and** $a < b$
shows $\textit{ith-digit}\ (A + a * b\hat{}i)\ i = a$
⟨*proof*⟩

lemma *current-term*:
fixes $A\ a\ c\ i$ :: *nat*
assumes $b{*}b\hat{}i\ \textit{dvd}\ A$ **and** $c < b\hat{}i$ **and** $a < b$
shows $\textit{ith-digit}\ (A + a{*}b\hat{}i + c)\ i = a$
⟨*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 :: $\textit{nat} \Rightarrow \textit{nat}$
assumes *eventually-zero* D **and** $m = (\sum i.\ D\ i * b\hat{}i)$ **and** $\bigwedge i.\ D\ i < b$
shows $D\ j = \textit{ith-digit}\ m\ j$
⟨*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 write, 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*) $_{LEbase\ b} = d + b * (d\text{-}list_{LEbase\ b})$

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** [a_0::*nat*, a_1] $_{LEbase\ b} = a_0 + a_1 * b$
⟨*proof*⟩

lemma *LEbase-three-digits* [*simp*]: **shows** [a_0::*nat*, a_1, a_2] $_{LEbase\ b} = a_0 + a_1 * b + a_2 * b\hat{}2$
⟨*proof*⟩

lemma *LEbase-closed-form*:
shows (*A* :: *nat list*) $_{LEbase\ b} = (\sum i < length\ A\ .\ A!i * b\hat{}i)$
⟨*proof*⟩

lemma *LEbase-concatenate*:
fixes *A D* :: *nat list* **and** *b* :: *nat*
shows $(A\ @\ D)_{LEbase\ b} = (A_{LEbase\ b}) + b\hat{}(length\ A) * (D_{LEbase\ b})$
⟨*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 $(\sum i.\ ith\text{-}digit\ m\ i\ *\ b\hat{}i) = (LEdigits\ m)_{LEbase\ b}$
⟨*proof*⟩

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

lemma *LElist-uniqueness*:
fixes *D* :: *nat list*
assumes ∀ *i* < *length D*. *D*!*i* < *b* **and** *D* = [] ∨ *last D* ≠ *0*
shows $LEdigits\ (D_{LEbase\ b}) = D$
⟨*proof*⟩

lemma *LE-digits-zero* [*simp*]: *LEdigits 0* = []
⟨*proof*⟩

lemma *LE-units-digit* [*simp*]:
assumes (*m*::*nat*) ∈ {*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* ($-_{base\ -}$ [*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** $[a_2{::}nat,\ a_1,\ a_0]_{\ base\ b} = a_2{*}b\widehat{\ }2 + a_1{*}b + a_0$
⟨*proof*⟩

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

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

context *digits-in-base*
begin

definition *digits* :: *nat* ⇒ *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 $(\sum i.\ ith\text{-}digit\ m\ i * b\widehat{\ }i) = (digits\ m)_{base\ b}$
⟨*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 $D_1, \ldots, D_n$ 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* :: *nat list*
assumes ∀ *d* ∈ *set D*. *d* < *b* **and** *D* = [] ∨ *D*!*0* ≠ *0*
shows *digits* (*D*$_{base\ b}$) = *D*
⟨*proof*⟩

We now prove some simplification rules (including a reccurrence 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 *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* ∈ {*0*<..<*b*}
shows *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* ≥ *b*
shows *digits m* = (*digits* (*m div b*)) @ [*m 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.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.