Digit Expansions

Jonas Bayer, Marco David, Abhik Pal and Benedikt Stock

May 26, 2022

Abstract

We formalize how a natural number a can be expanded as

$$a = \sum_{k=0}^{l} a_k b^k$$

for some base b and prove properties about functions that operate on such expansions. This includes the formalization of concepts such as digit shifts and carries. For a base that is a power of 2 we formalize the binary AND, binary orthogonality and binary masking of two natural numbers. This library on digit expansions builds the basis for the formalization of the DPRM theorem.

Contents

1 Digit functions

theory *Bits-Digits*
imports *Main*
begin

We define the n-th bit of a number in base 2 representation

definition *nth-bit* :: *nat* ⇒ *nat* ⇒ *nat* (**infix** ¡ *100*) **where**
nth-bit num k = (*num div* (*2* ^ *k*)) *mod 2*

as well as the n-th digit of a number in an arbitrary base

definition *nth-digit* :: *nat* ⇒ *nat* ⇒ *nat* ⇒ *nat* **where**
nth-digit num k base = (*num div* (*base* ^ *k*)) *mod base*

In base 2, the two definitions coincide.

lemma *nth-digit-base2-equiv*:*nth-bit a k* = *nth-digit a k* (*2*::*nat*)
⟨*proof*⟩

lemma *general-digit-base*:
assumes *t1* > *t2* **and** *b*>*1*
shows *nth-digit* (*a* ∗ *b*^*t1*) *t2 b* = *0*
⟨*proof*⟩

lemma *nth-bit-bounded*: *nth-bit a k* ≤ *1*
⟨*proof*⟩

lemma *nth-digit-bounded*: *b*>*1* ⟹ *nth-digit a k b* ≤ *b*−*1*
⟨*proof*⟩

lemma *obtain-smallest*: *P* (*n*::*nat*) ⟹ ∃ *k*≤*n*. *P k* ∧ (∀ *a*<*k*.¬(*P a*))
⟨*proof*⟩

1.1 Simple properties and equivalences

Reduce the *nth-digit* function to (¡) if the base is a power of 2

lemma *digit-gen-pow2-reduct*: *k*<*c* ⟹ (*nth-digit a t* (*2*^*c*)) ¡ *k* = *a* ¡ (*c*∗*t*+*k*)
⟨*proof*⟩

Show equivalence of numbers by equivalence of all their bits (digits)

lemma *aux-even-pow2-factor*: *a* > *0* ⟹ ∃ *k b*. ((*a*::*nat*) = (*2*^*k*) ∗ *b* ∧ *odd b*)
⟨*proof*⟩

lemma *aux0-digit-wise-equiv*:*a* > *0* ⟹ (∃ *k*. *nth-bit a k* = *1*)
⟨*proof*⟩

lemma *aux1-digit-wise-equiv*:(∀ *k*.(*nth-bit a k* = *0*)) ⟷ *a* = *0* (**is** *?P* ⟷ *?Q*)
⟨*proof*⟩

lemma *aux2-digit-wise-equiv*: $(\forall\, r{<}k.\ nth\text{-}bit\ a\ r = 0) \longrightarrow (a\ mod\ 2\hat{}k = 0)$
⟨*proof*⟩

lemma *digit-wise-equiv*: $(a = b) \longleftrightarrow (\forall\, k.\ nth\text{-}bit\ a\ k = nth\text{-}bit\ b\ k)$ (**is** *?P* $\longleftrightarrow$ *?Q*)
⟨*proof*⟩

Represent natural numbers in their binary expansion

lemma *aux3-digit-sum-repr*:
assumes $b < 2\hat{}r$
shows $(a{*}2\hat{}r + b)$ ¡ $r = (a{*}2\hat{}r)$ ¡ r
⟨*proof*⟩

lemma *aux2-digit-sum-repr*:
assumes $n < 2\hat{}c$ $r < c$
shows $(a{*}2\hat{}c{+}n)$ ¡ $r = n$ ¡ r
⟨*proof*⟩

lemma *aux1-digit-sum-repr*:
assumes $n < 2\hat{}c$ $r{<}c$
shows $(\sum k{<}c.((n$ ¡ $k){*}2\hat{}k))$ ¡ $r = n$ ¡ r
⟨*proof*⟩

lemma *digit-sum-repr*:
assumes $n < 2\hat{}c$
shows $n = (\sum k < c.((n$ ¡ $k) * 2\hat{}k))$
⟨*proof*⟩

lemma *digit-sum-repr-variant*:
$n = (\sum k{<}n.((nth\text{-}bit\ n\ k){*}2\hat{}k))$
⟨*proof*⟩

lemma *digit-sum-index-variant*:
$r{>}n \longrightarrow ((\sum k{<}\ n.((n$ ¡ $k){*}2\hat{}k)) = (\sum k{<}\ r.(n$ ¡ $k){*}2\hat{}k))$
⟨*proof*⟩

Digits are preserved under shifts

lemma *digit-shift-preserves-digits*:
assumes *b>1*
shows $nth\text{-}digit\ (b * y)\ (Suc\ t)\ b = nth\text{-}digit\ y\ t\ b$
⟨*proof*⟩

lemma *digit-shift-inserts-zero-least-siginificant-digit*:
assumes *t>0* **and** *b>1*
shows $nth\text{-}digit\ (1 + b * y)\ t\ b = nth\text{-}digit\ (b * y)\ t\ b$
⟨*proof*⟩

Represent natural numbers in their base-b digitwise expansion

lemma *aux3-digit-gen-sum-repr*:

assumes $d < b\hat{}r$ **and** $b > 1$
shows *nth-digit* $(a*b\hat{}r + d)\ r\ b =$ *nth-digit* $(a*b\hat{}r)\ r\ b$
⟨*proof*⟩

lemma *aux2-digit-gen-sum-repr*:
assumes $n < b\hat{}c\ r < c$
shows *nth-digit* $(a*b\hat{}c+n)\ r\ b =$ *nth-digit* $n\ r\ b$
⟨*proof*⟩

lemma *aux1-digit-gen-sum-repr*:
assumes $n < b\hat{}c\ r<c$ **and** $b>1$
shows *nth-digit* $(\sum k<c.((\text{nth-digit}\ n\ k\ b)*b\hat{}k))\ r\ b =$ *nth-digit* $n\ r\ b$
⟨*proof*⟩

lemma *aux-gen-b-factor*: $a > 0 \Longrightarrow b>1 \Longrightarrow \exists\, k\ c.\ ((a::nat) = (b\hat{}k) * c \wedge \neg(c\ mod\ b = 0))$
⟨*proof*⟩

lemma *aux0-digit-wise-gen-equiv*:
assumes $b>1$ **and** *a-geq-0*: $a > 0$
shows $(\exists\, k.$ *nth-digit* $a\ k\ b \neq 0)$
⟨*proof*⟩

lemma *aux1-digit-wise-gen-equiv*:
assumes $b>1$
shows $(\forall\, k.(\text{nth-digit}\ a\ k\ b = 0)) \longleftrightarrow a = 0$ (**is** $?P \longleftrightarrow ?Q$)
⟨*proof*⟩

lemma *aux2-digit-wise-gen-equiv*: $(\forall\, r<k.$ *nth-digit* $a\ r\ b = 0) \longrightarrow (a\ mod\ b\hat{}k = 0)$
⟨*proof*⟩

Two numbers are the same if and only if their digits are the same

lemma *digit-wise-gen-equiv*:
assumes $b>1$
shows $(x = y) \longleftrightarrow (\forall\, k.$ *nth-digit* $x\ k\ b =$ *nth-digit* $y\ k\ b)$ (**is** $?P \longleftrightarrow ?Q$)
⟨*proof*⟩

A number is equal to the sum of its digits multiplied by powers of two

lemma *digit-gen-sum-repr*:
assumes $n < b\hat{}c$ **and** $b>1$
shows $n = (\sum k < c.((\text{nth-digit}\ n\ k\ b) * b\hat{}k))$
⟨*proof*⟩

lemma *digit-gen-sum-repr-variant*:
assumes $b>1$
shows $n = (\sum k<n.((\text{nth-digit}\ n\ k\ b)*b\hat{}k))$
⟨*proof*⟩

lemma *digit-gen-sum-index-variant*:
assumes *b>1* **shows** *r>n* $\Longrightarrow$
$(\sum k< n.((nth\text{-}digit\ n\ k\ b\ )*b\hat{}k)) = (\sum k< r.(nth\text{-}digit\ n\ k\ b)*b\hat{}k)$
⟨*proof*⟩

nth-digit extracts coefficients from a base-b digitwise expansion

lemma *nth-digit-gen-power-series*:
fixes *c b k q*
defines $b \equiv 2\hat{}(Suc\ c)$
assumes *bound*: $\forall k.\ (f\ k) < b$
shows $nth\text{-}digit\ (\sum k{=}0..q.\ (f\ k) * b\hat{}k)\ t\ b = (if\ t{\leq}q\ then\ (f\ t)\ else\ 0)$
⟨*proof*⟩

Equivalence condition for the *nth-digit* function [1] (see equation 2.29)

lemma *digit-gen-equiv*:
assumes *b>1*
shows $d = nth\text{-}digit\ a\ k\ b \longleftrightarrow (\exists x.\exists y.(a = x * b\hat{}(k{+}1) + d{*}b\hat{}k + y \wedge d < b \wedge y < b\hat{}k))$
(**is** *?P* $\longleftrightarrow$ *?Q*)
⟨*proof*⟩

end
theory *Carries*
imports *Bits-Digits*
begin

2 Carries in base-b expansions

Some auxiliary lemmas

lemma *rev-induct*[*consumes 1*, *case-names base step*]:
fixes *i k* :: *nat*
assumes *le*: $i \leq k$
and *base*: *P k*
and *step*: $\bigwedge i.\ i \leq k \Longrightarrow P\ i \Longrightarrow P\ (i - 1)$
shows *P i*
⟨*proof*⟩

2.1 Definition of carry received at position k

When adding two numbers m and n, the carry is *introduced* at position 1 but is *received* at position 2. The function below accounts for the latter case.

```
      k: 6 5 4 3 2 1 0
      c:       1
  - - - - - - - - - - - -
      m:   1 0 1 0 1 0
```

```
    n:           1 1
----------------------
m + n: 0 1 0 1 1 0 0
```

definition *bin-carry* :: *nat* ⇒ *nat* ⇒ *nat* ⇒ *nat* **where**
bin-carry a b k = (*a mod 2^k* + *b mod 2^k*) *div 2^k*

Carry in the subtraction of two natural numbers

definition *bin-narry* :: *nat* ⇒ *nat* ⇒ *nat* ⇒ *nat* **where**
bin-narry a b k = (*if b mod 2^k* > *a mod 2^k then 1 else 0*)

Equivalent definition

definition *bin-narry2* :: *nat* ⇒ *nat* ⇒ *nat* ⇒ *nat* **where**
bin-narry2 a b k = ((*2^k* + *a mod 2^k* − *b mod 2^k*) *div 2^k* + *1*) *mod 2*

lemma *bin-narry-equiv*: *bin-narry a b c* = *bin-narry2 a b c*
⟨*proof*⟩

2.2 Properties of carries

lemma *div-sub*:
fixes *a b c* :: *nat*
shows (*a* − *b*) *div c* = (*if*(*a mod c* < *b mod c*) *then a div c* − *b div c* − *1 else a div c* − *b div c*)
⟨*proof*⟩

lemma *dif-digit-formula*:*a* ≥ *b* ⟶ (*a* − *b*)¡*k* = (*a*¡*k* + *b*¡*k* + *bin-narry a b k*) *mod 2*
⟨*proof*⟩

lemma *dif-narry-formula*:
a≥*b* ⟶ *bin-narry a b* (*k* + *1*) = (*if* (*a*¡*k* < *b*¡*k* + *bin-narry a b k*) *then 1 else 0*)
⟨*proof*⟩

lemma *sum-digit-formula*:(*a* + *b*)¡*k* =(*a*¡*k* + *b*¡*k* + *bin-carry a b k*) *mod 2*
⟨*proof*⟩

lemma *sum-carry-formula*:*bin-carry a b* (*k* + *1*) =(*a*¡*k* + *b*¡*k* + *bin-carry a b k*) *div 2*
⟨*proof*⟩

lemma *bin-carry-bounded*:
shows *bin-carry a b k* = *bin-carry a b k mod 2*
⟨*proof*⟩

lemma *carry-bounded*: *bin-carry a b k* ≤ *1*
⟨*proof*⟩

lemma *no-carry*:
(∀ *r*< *n*.((*nth-bit a r*) + (*nth-bit b r*) ≤ *1*)) ⟹
(*nth-bit* (*a* + *b*) *n*) = (*nth-bit a n* + *nth-bit b n*) *mod 2*
(**is** *?P* ⟹ *?Q n*)
⟨*proof*⟩

lemma *no-carry-mult-equiv*:(∀ *k*. *nth-bit a k* ∗ *nth-bit b k* = *0*) ⟷ (∀ *k*. *bin-carry a b k* = *0*)
(**is** *?P* ⟷ *?Q*)
⟨*proof*⟩

lemma *carry-digit-impl*: *bin-carry a b k* ≠ *0* ⟹ ∃ *r*<*k*. *a* ¡ *r* + *b* ¡ *r* = *2*
⟨*proof*⟩

end
theory *Binary-Operations*
imports *Bits-Digits Carries*
begin

3 Digit-wise Operations

3.1 Binary AND

fun *bitAND-nat* :: *nat* ⇒ *nat* ⇒ *nat* (**infix** &&& *64*) **where**
0 &&& *-* = *0* |
m &&& *n* = *2* ∗ ((*m div 2*) &&& (*n div 2*)) + (*m mod 2*) ∗ (*n mod 2*)

lemma *bitAND-zero*[*simp*]: *n* = *0* ⟹ *m* &&& *n* = *0*
⟨*proof*⟩

lemma *bitAND-1*: *a* &&& *1* = (*a mod 2*)
⟨*proof*⟩

lemma *bitAND-rec*: *m* &&& *n* = *2* ∗ ((*m div 2*) &&& (*n div 2*)) + (*m mod 2*) ∗ (*n mod 2*)
⟨*proof*⟩

lemma *bitAND-commutes*:*m* &&& *n* = *n* &&& *m*
⟨*proof*⟩

lemma *nth-digit-0*: *x* ≤ *1* ⟹ *nth-bit x 0* = *x* ⟨*proof*⟩

lemma *bitAND-zeroone*: *a* ≤ *1* ⟹ *b* ≤ *1* ⟹ *a* &&& *b* ≤ *1*
⟨*proof*⟩

lemma *aux1-bitAND-digit-mult*:
fixes *a b c* :: *nat*
shows $k > 0 \wedge a \bmod 2 = 0 \wedge b \leq 1 \Longrightarrow (a + b) \mathbin{div} 2\hat{}k = a \mathbin{div} 2\hat{}k$
⟨*proof*⟩

lemma *bitAND-digit-mult*:$(nth\text{-}bit\ (a\ \&\&\ b)\ k) = (nth\text{-}bit\ a\ k) * (nth\text{-}bit\ b\ k)$
⟨*proof*⟩

lemma *bitAND-single-bit-mult-equiv*: $a \leq 1 \Longrightarrow b \leq 1 \Longrightarrow a * b = a\ \&\&\ b$
⟨*proof*⟩

lemma *bitAND-mult-equiv*:
$(\forall k.\ (nth\text{-}bit\ c\ k) = (nth\text{-}bit\ a\ k) * (nth\text{-}bit\ b\ k)) \longleftrightarrow c = a\ \&\&\ b$ (**is** *?P* $\longleftrightarrow$ *?Q*)
⟨*proof*⟩

lemma *bitAND-linear*:
fixes *k*::*nat*
shows $(b < 2\hat{}k) \wedge (d < 2\hat{}k) \Longrightarrow (a * 2\hat{}k + b)\ \&\&\ (c * 2\hat{}k + d) = (a\ \&\&\ c) * 2\hat{}k + (b\ \&\&\ d)$
⟨*proof*⟩

3.2 Binary orthogonality

cf. [1] section 2.6.1 on "Binary orthogonality"

The following definition differs slightly from the one in the paper. However, we later prove the equivalence of the two definitions.

fun *orthogonal* :: *nat* => *nat* => *bool* (**infix** $\perp$ *49*) **where**
$(orthogonal\ a\ b) = (a\ \&\&\ b = 0)$

lemma *ortho-mult-equiv*: $a \perp b \longleftrightarrow (\forall k.\ (nth\text{-}bit\ a\ k) * (nth\text{-}bit\ b\ k) = 0)$ (**is** *?P* $\longleftrightarrow$ *?Q*)
⟨*proof*⟩

lemma *aux1-1-digit-lt-linear*:
assumes $b < 2\hat{}r\ k \geq r$
shows $bin\text{-}carry\ (a{*}2\hat{}r)\ b\ k = 0$
⟨*proof*⟩

lemma *aux1-digit-lt-linear*:
assumes $b < 2\hat{}r$ **and** $k \geq r$
shows $(a{*}2\hat{}r + b)\ ¡\ k = (a{*}2\hat{}r)\ ¡\ k$
⟨*proof*⟩

lemma *aux-digit-shift*: $(a * 2\hat{}t)\ ¡\ (l{+}t) = a\ ¡\ l$
⟨*proof*⟩

lemma *aux-digit-lt-linear*:
assumes *b*: $b < (2::nat)\hat{}t$
assumes *d*: $d < (2::nat)\hat{}t$
shows $(a * 2\hat{}t + b) \mathbin{¡} k \leq (c * 2\hat{}t + d) \mathbin{¡} k \longleftrightarrow ((a * 2\hat{}t) \mathbin{¡} k \leq (c * 2\hat{}t) \mathbin{¡} k \wedge b \mathbin{¡} k \leq d \mathbin{¡} k)$
⟨*proof*⟩

lemma *aux2-digit-lt-linear*:
fixes *a b c d t l* :: *nat*
shows $\exists k.\ (a * 2\hat{}t) \mathbin{¡} k \leq (c * 2\hat{}t) \mathbin{¡} k \longrightarrow a \mathbin{¡} l \leq c \mathbin{¡} l$
⟨*proof*⟩

lemma *aux3-digit-lt-linear*:
fixes *a b c d t k* :: *nat*
shows $\exists l.\ a \mathbin{¡} l \leq c \mathbin{¡} l \longrightarrow (a * 2\hat{}t) \mathbin{¡} k \leq (c * 2\hat{}t) \mathbin{¡} k$
⟨*proof*⟩

lemma *digit-lt-linear*:
fixes *a b c d t* :: *nat*
assumes *b*: $b < (2::nat)\hat{}t$
assumes *d*: $d < (2::nat)\hat{}t$
shows $(\forall k.\ (a * 2\hat{}t + b) \mathbin{¡} k \leq (c * 2\hat{}t + d) \mathbin{¡} k) \longleftrightarrow (\forall l.\ a \mathbin{¡} l \leq c \mathbin{¡} l \wedge b \mathbin{¡} l \leq d \mathbin{¡} l)$
⟨*proof*⟩

Sufficient bitwise (digitwise) condition for the non-strict standard order of natural numbers

lemma *digitwise-leq*:
assumes *b>1*
shows $\forall t.$ *nth-digit x t b* $\leq$ *nth-digit y t b* $\Longrightarrow x \leq y$
⟨*proof*⟩

3.3 Binary masking

Preliminary result on the standard non-strict of natural numbers

lemma *bitwise-leq*: $(\forall k.\ a \mathbin{¡} k \leq\ b \mathbin{¡} k) \longrightarrow a \leq b$
⟨*proof*⟩

cf. [1] section 2.6.2 on "Binary Masking"

Again, the equivalence to the definition there will be proved in a later lemma.

fun *masks* :: *nat* => *nat* => *bool* (**infix** $\preceq$ *49*) **where**
masks 0 - = True |
masks a b = ((*a div 2* $\preceq$ *b div 2*) $\wedge$ (*a mod 2* $\leq$ *b mod 2*))

lemma *masks-substr*: $a \preceq b \Longrightarrow (a\ div\ (2\hat{}k) \preceq b\ div\ (2\hat{}k))$
⟨*proof*⟩

lemma *masks-digit-leq*:$(a \preceq b) \Longrightarrow (nth\text{-}bit\ a\ k) \leq (nth\text{-}bit\ b\ k)$
⟨*proof*⟩

lemma *masks-leq-equiv*:$(a \preceq b) \longleftrightarrow (\forall k.\ (nth\text{-}bit\ a\ k) \leq (nth\text{-}bit\ b\ k))$ (**is** *?P* $\longleftrightarrow$ *?Q*)
⟨*proof*⟩

lemma *masks-leq*:$a \preceq b \longrightarrow a \leq b$
⟨*proof*⟩

lemma *mask-linear*:
fixes *a b c d t* :: *nat*
assumes *b*: $b < (2::nat)\hat{}t$
assumes *d*: $d < (2::nat)\hat{}t$
shows $((a * 2\hat{}t + b \preceq c * 2\hat{}t + d) \longleftrightarrow (a \preceq c \wedge b \preceq d))$ (**is** *?P* $\longleftrightarrow$ *?Q*)
⟨*proof*⟩

lemma *aux1-lm0241-pow2-up-bound*:$(\exists (p::nat).\ (a::nat) < 2\hat{}(Suc\ p))$
⟨*proof*⟩

lemma *aux2-lm0241-single-digit-binom*:
assumes $1 \geq (a::nat)$
assumes $1 \geq (b::nat)$
shows $\neg(a = 1 \wedge b = 1) \longleftrightarrow ((a + b)\ choose\ b) = 1$ (**is** *?P* $\longleftrightarrow$ *?Q*)
⟨*proof*⟩

lemma *aux3-lm0241-binom-bounds*:
assumes $1 \geq (m::nat)$
assumes $1 \geq (n::nat)$
shows $1 \geq m\ choose\ n$
⟨*proof*⟩

lemma *aux4-lm0241-prod-one*:
fixes $f::(nat \Rightarrow nat)$
assumes $(\forall x.\ (1 \geq f\ x))$
shows $(\prod k \leq n.\ (f\ k)) = 1 \longrightarrow (\forall k.\ k \leq n \longrightarrow f\ k = 1)$ (**is** *?P* $\longrightarrow$ *?Q*)
⟨*proof*⟩

lemma *aux5-lm0241*:
$(\forall i.\ (nth\text{-}bit\ (a + b)\ i)\ choose\ (nth\text{-}bit\ b\ i) = 1) \longrightarrow$
$\neg(nth\text{-}bit\ a\ i = 1 \wedge nth\text{-}bit\ b\ i = 1)$
(**is** *?P* $\longrightarrow$ *?Q i*)
⟨*proof*⟩

end

References

[1] Y. Matiyasevich. On Hilbert's tenth problem. In M. Lamoureux, editor, *PIMS Distinguished Chair Lectures*, volume 1. Pacific Institute for the Mathematical Sciences, 2000.