Proving a Property of XOR in PVS

Kai Engelhardt

February 10, 2023

A post on cs.stackexchange.com asked why the sequence of folds of prefixes of the natural numbers combined with XOR has this interesting "period" of 4. Here we prove fact that in PVS.

The claim is that, for all $n \in \mathbb{N}$,

$$\bigotimes_n = \begin{cases} n & \text{if } n \bmod 4 = 0 \\ 1 & \text{if } n \bmod 4 = 1 \\ n+1 & \text{if } n \bmod 4 = 2 \\ 0 & \text{if } n \bmod 4 = 3 \end{cases}$$

where $\bigotimes$ is defined inductively by

$$\bigotimes_0 = 0 \qquad\qquad \bigotimes_{k+1} = \bigotimes_k \otimes (k+1)$$

and $\otimes$ is bit-wise exclusive "or" on the binary encoding of its arguments. Where the lengths of the encodings differ we align to the right, that is, we combine the bits of matching significance.

Example 1 $7_{10} \otimes 8_{10} = 111_2 \otimes 1000_2 = \otimes\begin{array}{|c|c|c|c|}\hline & 1 & 1 & 1 \\ \hline 1 & 0 & 0 & 0 \\ \hline\end{array} = 1111_2 = 15_{10}$.

1 A PVS Encoding

```
xormod: THEORY
BEGIN
  l,n,m: VAR nat
  a,b,c: VAR nbit
  xor_bit(b,c): nbit =
   TABLE b, c
        %+---+---++
        |[ 0 | 1 ]|
     %---+---+---++
     | 0 | 0 | 1 ||
     %---+---+---++
     | 1 | 1 | 0 ||
     %---+---+---++
   ENDTABLE

```

```
  xor_bit_comm: LEMMA
    xor_bit(b,c) = xor_bit(c,b)
  xor_bit_assoc: LEMMA
    xor_bit(a,xor_bit(b,c)) = xor_bit(xor_bit(a,b),c)
  xor_bit_cancel: LEMMA
    xor_bit(b,b) = 0
  xor_bit_zero0: LEMMA
    xor_bit(b,0) = b
  xor_bit_zero1: LEMMA
    xor_bit(0,c) = c

  xor_nat(n,m): RECURSIVE nat =
    IF n = 0 THEN m
    ELSIF m = 0 THEN n
    ELSE LET n2 = ndiv(n,2), m2 = ndiv(m,2),
             n0 = rem(2)(n), m0 = rem(2)(m)
          IN xor_bit(n0,m0) + 2 * xor_nat(n2,m2)
    ENDIF
  MEASURE n+m

  xor_nat_comm: LEMMA
    xor_nat(n,m) = xor_nat(m,n)
  xor_nat_assoc: LEMMA
    xor_nat(l,xor_nat(n,m)) = xor_nat(xor_nat(l,n),m)
  xor_nat_cancel: LEMMA
    xor_nat(n,n) = 0
  xor_nat_zero0: LEMMA
    xor_nat(n,0) = n
  xor_nat_zero1: LEMMA
    xor_nat(0,m) = m
  xor_nat_one_even: LEMMA
    even?(n) IMPLIES xor_nat(n,1) = n+1
  xor_nat_one_odd: LEMMA
    odd?(n) IMPLIES xor_nat(n,1) = n-1
  xor_nat_succ_even: LEMMA
    even?(n) IMPLIES xor_nat(n,n+1) = 1
  % xor_nat_succ_odd: LEMMA
  %   odd?(n) IMPLIES xor_nat(n,n+1) = ??

  xor_iter(n): RECURSIVE nat =
    IF n = 0 THEN 0
    ELSE xor_nat(xor_iter(n-1),n)
    ENDIF
  MEASURE n

  xor_iter_prop: LEMMA
  LET m = rem(4)(n), x = xor_iter(n)
   IN
     TABLE
       %-------+---------++
       | m = 0 | x = n   ||
       %-------+---------++
       | m = 1 | x = 1   ||
       %-------+---------++
       | m = 2 | x = n+1 ||
       %-------+---------++
       | m = 3 | x = 0   ||
       %-------+---------++
     ENDTABLE
END xormod
```

That there's more than usual build-up of XOR-related lemmas is due to the fact that I only found bitvector version for bounded-size bitvectors in the PVS prelude. Here I thought I'd benefit from having arbitrary ones. It works quite easily anyway. The proof summary:

```
Proof summary for theory xormod
   xor_bit_TCC1.........................proved - complete   [shostak](0.01 s)
   xor_bit_TCC2.........................proved - complete   [shostak](0.01 s)
   xor_bit_TCC3.........................proved - complete   [shostak](0.01 s)
   xor_bit_comm.........................proved - complete   [shostak](0.02 s)
   xor_bit_assoc........................proved - complete   [shostak](0.08 s)
   xor_bit_cancel.......................proved - complete   [shostak](0.00 s)
   xor_bit_zero0........................proved - complete   [shostak](0.01 s)
   xor_bit_zero1........................proved - complete   [shostak](0.01 s)
   xor_nat_TCC1.........................proved - complete   [shostak](0.26 s)
   xor_nat_TCC2.........................proved - complete   [shostak](0.28 s)
   xor_nat_TCC3.........................proved - complete   [shostak](0.32 s)
   xor_nat_comm.........................proved - complete   [shostak](0.76 s)
   xor_nat_assoc........................untried              [Untried]( n/a s)
   xor_nat_cancel.......................proved - complete   [shostak](0.23 s)
   xor_nat_zero0........................proved - complete   [shostak](0.08 s)
   xor_nat_zero1........................proved - complete   [shostak](0.01 s)
   xor_nat_one_even.....................proved - complete   [shostak](1.33 s)
   xor_nat_one_odd......................proved - complete   [shostak](1.31 s)
   xor_nat_succ_even....................proved - complete   [shostak](0.97 s)
   xor_iter_TCC1........................proved - complete   [shostak](0.01 s)
   xor_iter_TCC2........................proved - complete   [shostak](0.00 s)
   xor_iter_prop_TCC1...................proved - complete   [shostak](0.08 s)
   xor_iter_prop........................proved - complete   [shostak](8.41 s)
   Theory xormod totals: 23 formulas, 22 attempted, 22 succeeded (14.21 s)
```

The only lemma not proved (labelled `untried` above) is not needed for the main result.