Bounding LCM with Triangles — Behind the Scenes

How the Proof becomes a Pearl

Hing-Lun Chan and Michael Norrish

College of Engineering and Computer Science
Australian National University

August 2016, ANU.

Recap

AKS mechanisation

PRIMES is in P

Manindra Agrawal Neeraj Kayal
Nitin Saxena*

Abstract

We present an unconditional deterministic polynomial-time algorithm that determines whether an input number is prime or composite.

We will need the following simple fact about the lcm of first m numbers (see, e.g., [Nai82] for a proof).

Lemma 3.1. *Let $LCM(m)$ denote the lcm of first m numbers. For $m \geq 7$:*

$$LCM(m) \geq 2^m.$$

Need to formalize this LCM lemma, but not using Nair's integral-sum.

Math Stack Exchange

Google search leads to Leibniz's Harmonic Triangle.

Is there a direct proof of this lcm identity?

26

7

The identity

$$(n+1)\operatorname{lcm}\left(\binom{n}{0},\binom{n}{1},\dots\binom{n}{n}\right) = \operatorname{lcm}(1,2,\dots n+1)$$

is probably not well-known. The only way I know how to prove it is by using Kummer's theorem that the power of p dividing $\binom{a+b}{a}$ is the number of carries needed to add a and b in base p. Is there a more direct proof, e.g. by showing that each side divides the other?

(number-theory) (binomial-coefficients)

share cite improve this question edited Aug 3 '10 at 8:04 asked Aug 3 '10 at 4:18

3 Answers

active oldest **votes**

19

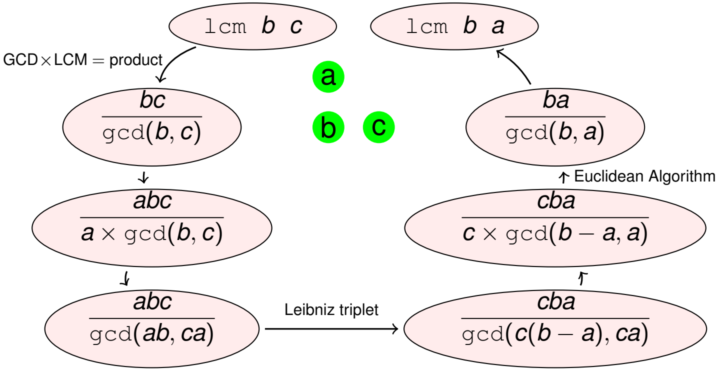
Consider *Leibniz harmonic triangle* — a table that is like «Pascal triangle reversed»: on it's sides lie numbers $\frac{1}{n}$ and each number is the sum of two beneath it (see the picture).

One can easily proove by induction that m-th number in n-th row of Leibniz triangle is $\frac{1}{\dots}$.

Key Property

Theorem (LCM Exchange)

For a Leibniz triplet $\{a, b, c\}$, $\texttt{lcm}\ b\ c = \texttt{lcm}\ b\ a.$



For a Leibniz triplet $\{a, b, c\}$, $\quad ab = c(b-a)$.

Clever Idea

	1					
	2	2				
	3	6	3			
	4	12	12	4		
	5	20	30	20	5	
	6	30	60	60	30	6
$6\times$	1	5	10	10	5	1

Theorem (Lower Bound for the LCM of any list ℓ)

For a list ℓ of positive numbers, $\texttt{SUM}\ \ell \le \texttt{LENGTH}\ \ell \times \texttt{list\_lcm}\ \ell$.

Clever Idea

	1					
	2	2				
	3	6	3			
	4	12	12	4		
	5	20	30	20	5	
	6	30	60	60	30	6
6$\times$	1	5	10	10	5	1

Theorem (Lower Bound for the LCM of any list ℓ)

For a list ℓ of positive numbers, $\texttt{SUM}\ \ell \leq \texttt{LENGTH}\ \ell \times \texttt{list\_lcm}\ \ell$.

- Applying theorem to vertical list . . . a disappointing lower bound.

Clever Idea

	1					
	2	2				
	3	6	3			
	4	12	12	4		
	5	20	30	20	5	
	6	30	60	60	30	6
6×	1	5	10	10	5	1

Theorem (Lower Bound for the LCM of any list ℓ)

For a list ℓ of positive numbers, $\texttt{SUM}\ \ell \leq \texttt{LENGTH}\ \ell \times \texttt{list\_lcm}\ \ell$.

- Applying theorem to vertical list . . . a disappointing lower bound.
- Applying theorem to horizontal list . . . ??

Clever Idea

	1					
	2	2				
	3	6	3			
	4	12	12	4		
	5	20	30	20	5	
	6	30	60	60	30	6
6$\times$	1	5	10	10	5	1

Theorem (Lower Bound for the LCM of any list ℓ)

For a list ℓ of positive numbers, $\texttt{SUM}\ \ell \leq \texttt{LENGTH}\ \ell \times \texttt{list\_lcm}\ \ell$.

- Applying theorem to vertical list . . . a disappointing lower bound.
- Applying theorem to horizontal list . . . ??
  This will work because both lists have the same LCM!

The Journey

Google: Hits and Misses

Google: Hits and Misses

- Google: “LCM lower bound”
  - May not get Q1442: Is there a Direct Proof of this LCM identity?

Google: Hits and Misses

- Google: "LCM lower bound"
  - May not get Q1442: Is there a Direct Proof of this LCM identity?
- Google: "LCM identity"
  - Lucky to find Q1442, giving only a sketch of the proof.

Google: Hits and Misses

- Google: “LCM lower bound”
  - May not get Q1442: Is there a Direct Proof of this LCM identity?
- Google: “LCM identity”
  - Lucky to find Q1442, giving only a sketch of the proof.
- Many more things to search for:
  - Leibniz harmonic triangle
  - LCM and triangle

Google: Hits and Misses

- Google: “LCM lower bound”
  - May not get Q1442: Is there a Direct Proof of this LCM identity?
- Google: “LCM identity”
  - Lucky to find Q1442, giving only a sketch of the proof.
- Many more things to search for:
  - Leibniz harmonic triangle — not much.
  - LCM and triangle — no specific match.

Google: Hits and Misses

- Google: "LCM lower bound"
  - May not get Q1442: Is there a Direct Proof of this LCM identity?
- Google: "LCM identity"
  - Lucky to find Q1442, giving only a sketch of the proof.
- Many more things to search for:
  - Leibniz harmonic triangle — not much.
  - LCM and triangle — no specific match.

At some point, need to stop surfing and DIY.

3 Answers

active oldest **votes**

19

Consider *Leibniz harmonic triangle* — a table that is like «Pascal triangle reversed»: on it's sides lie numbers $\frac{1}{n}$ and each number is the sum of two beneath it (see the picture).

One can easily proove by induction that m-th number in n-th row of Leibniz triangle is $\frac{1}{(n+1)\binom{n}{m}}$.

So LHS of our identity is just lcd of fractions in n-th row of the triangle.

But it's not hard to see that any such number is an integer linear combination of fractions on triangle's sides (i.e. $1/1, 1/2, \dots, 1/n$) — and vice versa. So LHS is equal to $lcd(1/1, \dots, 1/n)$ — and that is exactly RHS.

Induction Pattern

```
(* LCM Lemma

(n+1) lcm (C(n,0) to C(n,n)) = lcm (1 to (n+1))

m-th number in the n-th row of Leibniz triangle is:  1/ (n+1)C(n,m)

So LHS = lcd (1/1, 1/2, 1/3, ..., 1/n) = RHS = lcm (1,2,3, ..., (n+1)).

0-th row:                   1
1-st row:               1/2   1/2
2-nd row:            1/3   1/6   1/3
3-rd row:        1/4   1/12   1/12   1/4
4-th row: 1/5  1/20  1/30  1/20  1/5

4-th row: 1/5 C(4,m), C(4,m) = 1 4 6 4 1, hence 1/5 1/20 1/30 1/20 1/5
  lcd (1/5 1/20 1/30 1/20 1/5)
= lcm (5, 20, 30, 20, 5)
= lcm (5 C(4,0), 5 C(4,1), 5 C(4,2), 5 C(4,3), 5 C(4,4))
= 5 lcm (C(4,0), C(4,1), C(4,2), C(4,3), C(4,4))
```

Induction Pattern

```
(* LCM Lemma

(n+1) lcm (C(n,0) to C(n,n)) = lcm (1 to (n+1))

m-th number in the n-th row of Leibniz triangle is:  1/ (n+1)C(n,m)

So LHS = lcd (1/1, 1/2, 1/3, ..., 1/n) = RHS = lcm (1,2,3, ..., (n+1)).

0-th row:                 1
1-st row:             1/2   1/2
2-nd row:          1/3   1/6   1/3
3-rd row:      1/4   1/12   1/12   1/4
4-th row: 1/5   1/20   1/30   1/20   1/5

4-th row: 1/5 C(4,m), C(4,m) = 1 4 6 4 1, hence 1/5 1/20 1/30 1/20 1/5
  lcd (1/5 1/20 1/30 1/20 1/5)
= lcm (5, 20, 30, 20, 5)
= lcm (5 C(4,0), 5 C(4,1), 5 C(4,2), 5 C(4,3), 5 C(4,4))
= 5 lcm (C(4,0), C(4,1), C(4,2), C(4,3), C(4,4))

```

- How to prove the identity by induction? Need a pattern.

Induction Pattern

```
(* LCM Lemma

(n+1) lcm (C(n,0) to C(n,n)) = lcm (1 to (n+1))

m-th number in the n-th row of Leibniz triangle is:  1/ (n+1)C(n,m)

So LHS = lcd (1/1, 1/2, 1/3, ..., 1/n) = RHS = lcm (1,2,3, ..., (n+1)).

0-th row:              1
1-st row:           1/2  1/2
2-nd row:         1/3  1/6  1/3
3-rd row:     1/4  1/12  1/12  1/4
4-th row: 1/5  1/20  1/30  1/20  1/5

4-th row: 1/5 C(4,m), C(4,m) = 1 4 6 4 1, hence 1/5 1/20 1/30 1/20 1/5
  lcd (1/5 1/20 1/30 1/20 1/5)
= lcm (5, 20, 30, 20, 5)
= lcm (5 C(4,0), 5 C(4,1), 5 C(4,2), 5 C(4,3), 5 C(4,4))
= 5 lcm (C(4,0), C(4,1), C(4,2), C(4,3), C(4,4))
```

- How to prove the identity by induction? Need a pattern.
- Assuming the identity, does it lead to the lower bound?

Finding Pattern

```
Theorem: In the Multiples Triangle, the vertical-lcm = horizontal-lcm.
i.e.     lcm (1, 2, 3) = lcm (3, 6, 3) = 6
         lcm (1, 2, 3, 4) = lcm (4, 12, 12, 4) = 12
         lcm (1, 2, 3, 4, 5) = lcm (5, 20, 30, 20, 5) = 60
         lcm (1, 2, 3, 4, 5, 6) = lcm (6, 30, 60, 60, 30, 6) = 60
Proof: With reference to Leibniz's Triangle, note: term = left-up - left
  lcm (5, 20, 30, 20, 5)
= lcm (5, 20, 30)                    by reduce repetition
= lcm (5, d(1/20), d(1/30))          by denominator of fraction
= lcm (5, d(1/4 - 1/5), d(1/30))     by term = left-up - left
= lcm (5, lcm(4, 5), d(1/12 - 1/20))     by denominator of fraction subtraction
= lcm (5, 4, lcm(12, 20))                by lcm (a, lcm (a, b)) = lcm (a, b)
= lcm (5, 4, lcm(d(1/12), d(1/20)))      to fraction again
= lcm (5, 4, lcm(d(1/3 - 1/4), d(1/4 - 1/5)))   by Leibniz's Triangle
= lcm (5, 4, lcm(lcm(3,4),     lcm(4,5)))     by fraction subtraction denominator
= lcm (5, 4, lcm(3, 4, 5))                    by lcm merge
= lcm (5, 4, 3)                               merge again
= lcm (5, 4, 3, 2)                            by lcm include factor (!!!)
= lcm (5, 4, 3, 2, 1)                         by lcm include 1
```

A sample of my investigation, by examples.

Promising Result

```

  lcm (1 to 5)                = 1x2x3x4x5/2 = 60
= 5 lcm (1 4 6 4 1)           = 5 x 12
=  lcm (1 4 6 4 1)            --> unfold 5x to add 5 times
 + lcm (1 4 6 4 1)
 + lcm (1 4 6 4 1)
 + lcm (1 4 6 4 1)
 + lcm (1 4 6 4 1)
>= 1 + 4 + 6 + 4 + 1          --> pick one of each 5 C(n,m), i.e. diagonal
= (1 + 1)^4                   --> fold back binomial
= 2^4                         = 16

Actually, can take 5 lcm (1 4 6 4 1) >= 5 x 6 = 30,
but this will need estimation of C(n, n/2), or C(2n,n), i.e. Stirling's formula.

Theorem: lcm (x y z) >= x  or lcm (x y z) >= y  or lcm (x y z) >= z

```

Figure out that the LCM identity leads to the desired lower bound.

Hit an Idea

```
(* The Idea:
                                                               b
Actually, lcm a b = lcm b c = lcm c a      for    a c  in Leibniz Triangle.
The only relationship is: c = ab/(a - b), or ab = c(a - b).

Is this a theorem:  ab = c(a - b)  ==> lcm a b = lcm b c = lcm c a
Or in fractions,   1/c = 1/b - 1/a ==> lcm a b = lcm b c = lcm c a ?

  lcm a b
= a b / (gcd a b)
= c(a - b) / (gcd a (a - b))
= ac(a - b) / gcd a (a-b) / a
= lcm (a (a-b)) c / a
= lcm (ca c(a-b)) / a
= lcm (ca ab) / a
= lcm b c

  lcm b c
= b c / gcd b c
= a b c / gcd a*b a*c
= a b c / gcd c*(a-b) c*a
= a b / gcd (a-b) a
= a b / gcd b a
= lcm (a b)
= lcm a b
```

Focus on a triplet . . .

Hit an Idea

```
(* The Idea:
                                                           b
Actually, lcm a b = lcm b c = lcm c a     for   a c  in Leibniz Triangle.
The only relationship is: c = ab/(a - b), or ab = c(a - b).

Is this a theorem:  ab = c(a - b)  ==> lcm a b = lcm b c = lcm c a
Or in fractions,   1/c = 1/b - 1/a ==> lcm a b = lcm b c = lcm c a ?

  lcm a b
= a b / (gcd a b)
= c(a - b) / (gcd a (a - b))
= ac(a - b) / gcd a (a-b) / a
= lcm (a (a-b)) c / a
= lcm (ca c(a-b)) / a
= lcm (ca ab) / a
= lcm b c

  lcm b c
= b c / gcd b c
= a b c / gcd a*b a*c
= a b c / gcd c*(a-b) c*a
= a b / gcd (a-b) a
= a b / gcd b a
= lcm (a b)
= lcm a b
```

Focus on a triplet … hope: $\texttt{lcm}\ a\ b = \texttt{lcm}\ b\ c = \texttt{lcm}\ c\ a$.

Voliá

```
(* The Idea:
                                                    b
Actually, lcm a b = lcm b c = lcm c a     for   a c  in Leibniz Triangle.
The only relationship is: c = ab/(a - b), or ab = c(a - b).

Is this a theorem:  ab = c(a - b)  ==> lcm a b = lcm b c = lcm c a
Or in fractions,   1/c = 1/b - 1/a ==> lcm a b = lcm b c = lcm c a ?


  lcm a c
= a c / gcd a c
= a b c / gcd b*a b*c
= a b c / gcd c*(a-b) b*c
= a b / gcd (a-b) b
= a b / gcd a b
= lcm a b

Yes!

This is now in LCM_EXCHANGE:
val it = |- !a b c. (a * b = c * (a - b)) ==> (lcm a b = lcm a c): thm
*)
```

Success!

Polishing

Done and Dusted

- Once the key is proved (SourceTree #1200), goal is within reach.
- Had the picture of path transform, zig-zag and wriggle, for induction.
- Just establish the LCM lower bound by brute-force induction (#1211).

Done and Dusted

- Once the key is proved (SourceTree #1200), goal is within reach.
- Had the picture of path transform, zig-zag and wriggle, for induction.
- Just establish the LCM lower bound by brute-force induction (#1211).

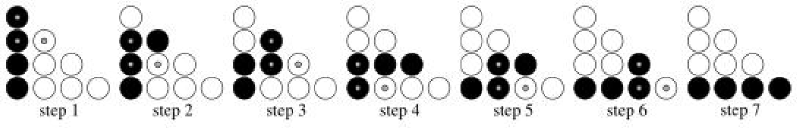
```
   Transform from Vertical LCM to Horizontal LCM:
   leibniz_lcm_shift_one      |- !n k. k <= n ==>
                                 (lcm (list_lcm (TAKE (SUC k) (leibniz_horizontal (SUC n))))
                                      (list_lcm (DROP k (leibniz_horizontal n))) =
                                  lcm (list_lcm (TAKE (SUC (SUC k)) (leibniz_horizontal (SUC n))))
                                      (list_lcm (DROP (SUC k) (leibniz_horizontal n))))
   leibniz_lcm_shift          |- !n k. k <= SUC n ==>
                                 (lcm (list_lcm (TAKE (SUC k) (leibniz_horizontal (SUC n))))
                                      (list_lcm (DROP k (leibniz_horizontal n))) =
                                  lcm (SUC (SUC n)) (list_lcm (leibniz_horizontal n)))
   leibniz_horizontal_lcm     |- !n. list_lcm (leibniz_horizontal (SUC n)) =
                                     lcm (SUC (SUC n)) (list_lcm (leibniz_horizontal n))
   leibniz_lcm_property       |- !n. list_lcm (leibniz_vertical n) = list_lcm (leibniz_horizontal n)

   Binomial Horizontal List:
   binomial_horizontal_def    |- !n. binomial_horizontal n = GENLIST (binomial n) (SUC n)
   binomial_horizontal_0      |- binomial_horizontal 0 = [1]
   binomial_horizontal_len    |- !n. LENGTH (binomial_horizontal n) = n + 1
   binomial_horizontal_pos    |- !n. EVERY (\x. 0 < x) (binomial_horizontal n)
   binomial_horizontal_sum    |- !n. SUM (binomial_horizontal n) = 2 ** n

   Lower Bound of Leibniz LCM:
   leibniz_alt                |- !n. leibniz n = (\k. (n + 1) * k) o binomial n
   leibniz_horizontal_alt     |- !n. leibniz_horizontal n = MAP (\k. (n + 1) * k) (binomial_horizontal n)
   leibniz_horizontal_lcm_alt          |- !n. list_lcm (leibniz_horizontal n) =
                                              (n + 1) * list_lcm (binomial_horizontal n)
   leibniz_horizontal_lcm_lower_bound  |- !n. 2 ** n <= list_lcm (leibniz_horizontal n)
   leibniz_vertical_lcm_lower_bound    |- !n. 2 ** n <= list_lcm (leibniz_vertical n)
*)
```

Back in Spotlight

- Decide to submit a paper to ITP2016 (a fortnight before deadline).
- Pick this LCM result for the category “Proof Pearl”.
- Use a picture to illustrate the path transform steps.

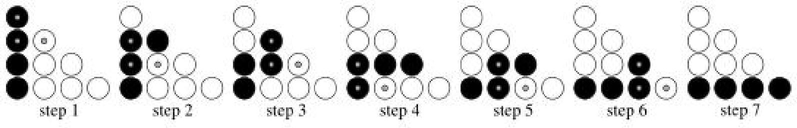
Back in Spotlight

- Decide to submit a paper to ITP2016 (a fortnight before deadline).
- Pick this LCM result for the category “Proof Pearl”.
- Use a picture to illustrate the path transform steps.



Back in Spotlight

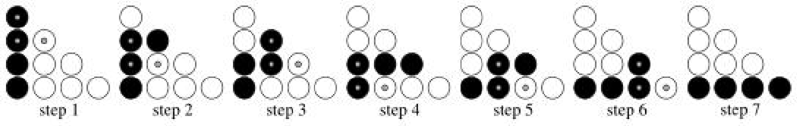
- Decide to submit a paper to ITP2016 (a fortnight before deadline).
- Pick this LCM result for the category “Proof Pearl”.
- Use a picture to illustrate the path transform steps.



- A good picture, but the proof script is bad — heaps of induction.

Back in Spotlight

- Decide to submit a paper to ITP2016 (a fortnight before deadline).
- Pick this LCM result for the category “Proof Pearl”.
- Use a picture to illustrate the path transform steps.



- A good picture, but the proof script is bad — heaps of induction.
- Realize that zig-zags and wriggles are implicit in current proofs.
- Replace brute-force induction bt explicit zig-zag and wriggle paths.

Major Changes

- Formalize in HOL4: path transform, zig-zag and wriggle.
- Reformulate the proofs based on such concepts (#1531).

Major Changes

- Formalize in HOL4: path transform, zig-zag and wriggle.
- Reformulate the proofs based on such concepts (#1531).

```

   Using Triplet and Paths:
   leibniz_zigzag_def     |- !path1 path2. path1 zigzag path2 <=>
                             ?n k part1 part2. (path1 = part1 ++ [tri_b] ++ [tri_a] ++ part2) /\
                                               (path2 = part1 ++ [tri_b] ++ [tri_c] ++ part2)
   leibniz_wriggle_def    |- !path1 path2. path1 wriggle path2 <=>
                             ?m f. (path1 = f 0) /\ (path2 = f m) /\ !k. k < m ==> f k zigzag f (SUC k)
   leibniz_lcm_triple     |- !n k. lcm tri_b tri_a = lcm tri_b tri_c
   list_lcm_zigzag        |- !path1 path2. path1 zigzag path2 ==> (list_lcm path1 = list_lcm path2)
   list_lcm_wriggle       |- !path1 path2. path1 wriggle path2 ==> (list_lcm path1 = list_lcm path2)
   leibniz_zigzag_wriggle    |- !path1 path2. path1 zigzag path2 ==> path1 wriggle path2
   leibniz_zigzag_tail       |- !path1 path2. path1 zigzag path2 ==> !x. [x] ++ path1 zigzag [x] ++ path2
   leibniz_wriggle_tail      |- !path1 path2. path1 wriggle path2 ==> !x. [x] ++ path1 wriggle [x] ++ path2
   leibniz_horizontal_wriggle
                          |- !n. [SUC (SUC n)] ++ leibniz_horizontal n wriggle leibniz_horizontal (SUC n)

   leibniz_up_0           |- leibniz_up 0 = [1]
   leibniz_up_len         |- !n. LENGTH (leibniz_up n) = SUC n
   leibniz_up_cons        |- !n. leibniz_up (SUC n) = SUC (SUC n)::leibniz_up n
   leibniz_triplet_0      |- leibniz_up 1 zigzag leibniz_horizontal 1
   leibniz_up_wriggle_horizontal  |- !n. 0 < n ==> leibniz_up n wriggle leibniz_horizontal n
   leibniz_lcm_property   |- !n. list_lcm (leibniz_vertical n) = list_lcm (leibniz_horizontal n)
*)
```

Major Changes

- Formalize in HOL4: path transform, zig-zag and wriggle.
- Reformulate the proofs based on such concepts (#1531).
- A 12-page draft, with wonderful diagrams, tables and proofs.

```

  Using Triplet and Paths:
  leibniz_zigzag_def    |- !path1 path2. path1 zigzag path2 <=>
                           ?n k part1 part2. (path1 = part1 ++ [tri_b] ++ [tri_a] ++ part2) /\
                                             (path2 = part1 ++ [tri_b] ++ [tri_c] ++ part2)
  leibniz_wriggle_def   |- !path1 path2. path1 wriggle path2 <=>
                           ?m f. (path1 = f 0) /\ (path2 = f m) /\ !k. k < m ==> f k zigzag f (SUC k)
  leibniz_lcm_triple    |- !n k. lcm tri_b tri_a = lcm tri_b tri_c
  list_lcm_zigzag       |- !path1 path2. path1 zigzag path2 ==> (list_lcm path1 = list_lcm path2)
  list_lcm_wriggle      |- !path1 path2. path1 wriggle path2 ==> (list_lcm path1 = list_lcm path2)
  leibniz_zigzag_wriggle   |- !path1 path2. path1 zigzag path2 ==> path1 wriggle path2
  leibniz_zigzag_tail      |- !path1 path2. path1 zigzag path2 ==> !x. [x] ++ path1 zigzag [x] ++ path2
  leibniz_wriggle_tail     |- !path1 path2. path1 wriggle path2 ==> !x. [x] ++ path1 wriggle [x] ++ path2
  leibniz_horizontal_wriggle
                        |- !n. [SUC (SUC n)] ++ leibniz_horizontal n wriggle leibniz_horizontal (SUC n)

  leibniz_up_0          |- leibniz_up 0 = [1]
  leibniz_up_len        |- !n. LENGTH (leibniz_up n) = SUC n
  leibniz_up_cons       |- !n. leibniz_up (SUC n) = SUC (SUC n)::leibniz_up n
  leibniz_triplet_0     |- leibniz_up 1 zigzag leibniz_horizontal 1
  leibniz_up_wriggle_horizontal  |- !n. 0 < n ==> leibniz_up n wriggle leibniz_horizontal n
  leibniz_lcm_property  |- !n. list_lcm (leibniz_vertical n) = list_lcm (leibniz_horizontal n)
*)
```

Final Touch

My supervisor's masterstrokes:

Final Touch

My supervisor's masterstrokes:

- Cut away half of the draft, keeping only 3 proofs (so 6 pages).

Final Touch

My supervisor's masterstrokes:

- Cut away half of the draft, keeping only 3 proofs (so 6 pages).
- Re-package diagrams and tables side-by-side, use explicit triplet.

Final Touch

My supervisor's masterstrokes:

- Cut away half of the draft, keeping only 3 proofs (so 6 pages).
- Re-package diagrams and tables side-by-side, use explicit triplet.
- Wriggle is the reflexive transitive closure (RTC) of zig-zag (#1567).

Final Touch

My supervisor's masterstrokes:

- Cut away half of the draft, keeping only 3 proofs (so 6 pages).
- Re-package diagrams and tables side-by-side, use explicit triplet.
- Wriggle is the reflexive transitive closure (RTC) of zig-zag (#1567).

```

   Wriggle Paths in Leibniz Triangle (old):
   leibniz_old_wriggle_def      |- !p1 p2. p1 old_wriggle p2 <=>
                                   ?m f. (p1 = f 0) /\ (p2 = f m) /\ !k. k < m ==> f k zigzag f (SUC k)
   list_lcm_old_wriggle         |- !p1 p2. p1 old_wriggle p2 ==> (list_lcm p1 = list_lcm p2)
   leibniz_zigzag_old_wriggle   |- !p1 p2. p1 zigzag p2 ==> p1 old_wriggle p2
   leibniz_old_wriggle_tail     |- !p1 p2. p1 old_wriggle p2 ==> !x. [x] ++ p1 old_wriggle [x] ++ p2
   leibniz_old_wriggle_trans    |- !p1 p2 p3. p1 old_wriggle p2 /\ p2 old_wriggle p3 ==> p1 old_wriggle p3
   leibniz_horizontal_old_wriggle    |- !n. [leibniz (n + 1) 0] ++ leibniz_horizontal n old_wriggle
                                             leibniz_horizontal (n + 1)

   Wriggle Paths in Leibniz Triangle (new):
   list_lcm_wriggle          |- !p1 p2. p1 wriggle p2 ==> (list_lcm p1 = list_lcm p2)
   leibniz_zigzag_wriggle    |- !p1 p2. p1 zigzag p2 ==> p1 wriggle p2
   leibniz_wriggle_tail      |- !p1 p2. p1 wriggle p2 ==> !x. [x] ++ p1 wriggle [x] ++ p2
   leibniz_wriggle_trans     |- !p1 p2 p3. p1 wriggle p2 /\ p2 wriggle p3 ==> p1 wriggle p3

   Back to Milestone Theorem:
   leibniz_triplet_0         |- leibniz_up 1 zigzag leibniz_horizontal 1
   leibniz_up_old_wriggle_horizontal   |- !n. 0 < n ==> leibniz_up n old_wriggle leibniz_horizontal n
   leibniz_lcm_property      |- !n. list_lcm (leibniz_vertical n) = list_lcm (leibniz_horizontal n)

*)
```

Final Touch

My supervisor's masterstrokes:

- Cut away half of the draft, keeping only 3 proofs (so 6 pages).
- Re-package diagrams and tables side-by-side, use explicit triplet.
- Wriggle is the reflexive transitive closure (RTC) of zig-zag (#1567).
- Last day: can't complete RTC induction. Help!

```

   Wriggle Paths in Leibniz Triangle (old):
   leibniz_old_wriggle_def      |- !p1 p2. p1 old_wriggle p2 <=>
                                   ?m f. (p1 = f 0) /\ (p2 = f m) /\ !k. k < m ==> f k zigzag f (SUC k)
   list_lcm_old_wriggle         |- !p1 p2. p1 old_wriggle p2 ==> (list_lcm p1 = list_lcm p2)
   leibniz_zigzag_old_wriggle   |- !p1 p2. p1 zigzag p2 ==> p1 old_wriggle p2
   leibniz_old_wriggle_tail     |- !p1 p2. p1 old_wriggle p2 ==> !x. [x] ++ p1 old_wriggle [x] ++ p2
   leibniz_old_wriggle_trans    |- !p1 p2 p3. p1 old_wriggle p2 /\ p2 old_wriggle p3 ==> p1 old_wriggle p3
   leibniz_horizontal_old_wriggle   |- !n. [leibniz (n + 1) 0] ++ leibniz_horizontal n old_wriggle
                                          leibniz_horizontal (n + 1)

   Wriggle Paths in Leibniz Triangle (new):
   list_lcm_wriggle          |- !p1 p2. p1 wriggle p2 ==> (list_lcm p1 = list_lcm p2)
   leibniz_zigzag_wriggle    |- !p1 p2. p1 zigzag p2 ==> p1 wriggle p2
   leibniz_wriggle_tail      |- !p1 p2. p1 wriggle p2 ==> !x. [x] ++ p1 wriggle [x] ++ p2
   leibniz_wriggle_trans     |- !p1 p2 p3. p1 wriggle p2 /\ p2 wriggle p3 ==> p1 wriggle p3

   Back to Milestone Theorem:
   leibniz_triplet_0         |- leibniz_up 1 zigzag leibniz_horizontal 1
   leibniz_up_old_wriggle_horizontal  |- !n. 0 < n ==> leibniz_up n old_wriggle leibniz_horizontal n
   leibniz_lcm_property      |- !n. list_lcm (leibniz_vertical n) = list_lcm (leibniz_horizontal n)

*)
```

Reviews

Review #1, Expertise: high

This paper presents a "proof pearl", a short and clever proof that $2^n \leq lcm(1, ..., n+1)$. This is not a trivial result: Nair's proof of this fact was published in 1982, and Google search reveals some recent strengthenings and generalizations, but it seems that there is no published elementary proof of this fact.

Review #1, Expertise: high

This paper presents a "proof pearl", a short and clever proof that $2^n \leq lcm(1, ..., n + 1)$. This is not a trivial result: Nair's proof of this fact was published in 1982, and Google search reveals some recent strengthenings and generalizations, but it seems that there is no published elementary proof of this fact.

[...], the authors have provided an elegant proof of an interesting result, and have formalized it. It certainly fits the description of a proof pearl.

Review #1, Expertise: high

This paper presents a "proof pearl", a short and clever proof that $2^n \leq lcm(1, ..., n+1)$*. This is not a trivial result: Nair's proof of this fact was published in 1982, and Google search reveals some recent strengthenings and generalizations, but it seems that there is no published elementary proof of this fact.*

[...], the authors have provided an elegant proof of an interesting result, and have formalized it. It certainly fits the description of a proof pearl.

The wording of Theorem 5 *is confusing. [...] How about saying this: [...]*

The reference to the "unrolling" in Section 5 *makes it mysterious, and the proof is needlessly baroque. The argument is simply this: [...]*

Review #2, Expertise: medium

The authors describe a (mechanised) proof of a number-theoretic fact: $2^n \leq lcm(1, ..., n+1)$*. The proof is not new, but the paper is advertised as a pearl.*

Review #2, Expertise: medium

The authors describe a (mechanised) proof of a number-theoretic fact: $2^n \leq lcm(1, ..., n+1)$*. The proof is not new, but the paper is advertised as a pearl.*

In the past I have reviewed several papers that were advertised as pearls, but that in my opinion were not pearls. That is not the case with this paper. I found the text engaging, and easy to follow. The proof is non-trivial, but the authors made it easy to understand for me, and I thought that the mechanisation was presented at a suitable level of detail.

Review #2, Expertise: medium

The authors describe a (mechanised) proof of a number-theoretic fact: $2^n \leq lcm(1, ..., n+1)$*. The proof is not new, but the paper is advertised as a pearl.*

In the past I have reviewed several papers that were advertised as pearls, but that in my opinion were not pearls. That is not the case with this paper. I found the text engaging, and easy to follow. The proof is non-trivial, but the authors made it easy to understand for me, and I thought that the mechanisation was presented at a suitable level of detail.

I strongly recommend the paper for publication.

Review #3, Expertise: medium

This proof pearl shows a lower bound for the least common multiple of the first n integers [...]

Review #3, Expertise: medium

This proof pearl shows a lower bound for the least common multiple of the first n integers [...]

Although the inequality is quite specific, this paper demonstrates that it is worth to search for elegant proofs rather than to apply the golden hammer of a complicated theory. Indeed, the formalised proof is very elementary compared to the published proofs I know of. The authors have done a good job of bringing together the proof ingredients (which have been known) and explaining the proof idea.

Review #3, Expertise: medium

This proof pearl shows a lower bound for the least common multiple of the first n integers [...]

Although the inequality is quite specific, this paper demonstrates that it is worth to search for elegant proofs rather than to apply the golden hammer of a complicated theory. Indeed, the formalised proof is very elementary compared to the published proofs I know of. The authors have done a good job of bringing together the proof ingredients (which have been known) and explaining the proof idea.

In summary, I think that this paper makes a nice proof pearl, and I therefore recommend acceptance.

Epilog

Conclusion

This talk is dedicated to

Michael Norrish,

my supervisor.

- Scripts
  `https://bitbucket.org/jhlchan/hol/src/`
  subfolder: `algebra/lib`.
- Paper
  `https://bitbucket.org/jhlchan/hol/downloads`