

Mersenne primes and the Lucas–Lehmer test

Manuel Eberl

May 16, 2026

Abstract

This article provides formal proofs of basic properties of Mersenne numbers, i. e. numbers of the form $2^n - 1$, and especially of Mersenne primes. In particular, an efficient, verified, and executable version of the Lucas–Lehmer test is developed. This test decides primality for Mersenne numbers in time polynomial in n .

Contents

1	Auxiliary material	2
1.1	Auxiliary number-theoretic material	2
1.2	Auxiliary algebraic material	3
2	The Lucas–Lehmer test	4
2.1	General properties of Mersenne numbers and Mersenne primes	4
2.2	The Lucas–Lehmer sequence	6
2.3	The ring $\mathbb{Z}[\sqrt{3}]$	6
2.4	The ring $(\mathbb{Z}/m\mathbb{Z})[\sqrt{3}]$	7
2.5	$\mathbb{Z}[\sqrt{3}]$ as a subring of \mathbb{R}	9
2.6	The canonical homomorphism $\mathbb{Z}[\sqrt{3}] \rightarrow (\mathbb{Z}/m\mathbb{Z})[\sqrt{3}]$	10
2.7	Correctness of the Lucas–Lehmer test	11
2.8	A first executable version Lucas–Lehmer test	12
3	Efficient code for testing Mersenne primes	12
3.1	Efficient computation of remainders modulo a Mersenne number	13
3.2	Efficient code for the Lucas–Lehmer sequence	14
3.3	Code for the Lucas–Lehmer test	15
3.4	Examples	15

1 Auxiliary material

```
theory Lucas-Lehmer-Auxiliary
imports
  HOL-Algebra.Ring
  Probabilistic-Prime-Tests.Jacobi-Symbol
begin
```

1.1 Auxiliary number-theoretic material

```
lemma congD:  $[a = b] \text{ (mod } n) \implies a \text{ mod } n = b \text{ mod } n$ 
  <proof>
```

```
lemma eval-coprime:
   $(b :: 'a :: euclidean-semiring-gcd) \neq 0 \implies \text{coprime } a \ b \longleftrightarrow \text{coprime } b \ (a \text{ mod } b)$ 
  <proof>
```

```
lemma two-power-odd-mod-12:
  assumes odd n n > 1
  shows  $[2 ^ n = 8] \text{ (mod } (12 :: nat))$ 
  <proof>
```

```
lemma Legendre-3-right:
  fixes p :: nat
  assumes p: prime p p > 3
  shows  $p \text{ mod } 12 \in \{1, 5, 7, 11\}$  and Legendre p 3 = (if p mod 12  $\in \{1, 7\}$ 
  then 1 else -1)
  <proof>
```

```
lemma Legendre-3-left:
  fixes p :: nat
  assumes p: prime p p > 3
  shows Legendre 3 p = (if p mod 12  $\in \{1, 11\}$  then 1 else -1)
  <proof>
```

```
lemma supplement2-Legendre':
  assumes prime p p  $\neq 2$ 
  shows Legendre 2 p = (if p mod 8 = 1  $\vee$  p mod 8 = 7 then 1 else -1)
  <proof>
```

```
lemma little-Fermat-nat:
  fixes a :: nat
  assumes prime p  $\neg p \text{ dvd } a$ 
  shows  $[a ^ p = a] \text{ (mod } p)$ 
  <proof>
```

```
lemma little-Fermat-int:
  fixes a :: int and p :: nat
  assumes prime p  $\neg p \text{ dvd } a$ 
  shows  $[a ^ p = a] \text{ (mod } p)$ 
```

<proof>

lemma *prime-dvd-choose*:

assumes $0 < k < p$ *prime* p

shows $p \text{ dvd } (p \text{ choose } k)$

<proof>

lemma *prime-natD*:

assumes *prime* $(p :: \text{nat})$ $a \text{ dvd } p$

shows $a = 1 \vee a = p$

<proof>

lemma *not-prime-imp-ex-prod-nat*:

assumes $m > 1$ \neg *prime* $(m :: \text{nat})$

shows $\exists n k. m = n * k \wedge 1 < n \wedge n < m \wedge 1 < k \wedge k < m$

<proof>

1.2 Auxiliary algebraic material

lemma (*in group*) *ord-eqI-prime-factors*:

assumes $\bigwedge p. p \in \text{prime-factors } n \implies x [\wedge] (n \text{ div } p) \neq \mathbf{1}$ **and** $x [\wedge] n = \mathbf{1}$

assumes $x \in \text{carrier } G$ $n > 0$

shows $\text{group.ord } G x = n$

<proof>

lemma (*in monoid*) *pow-nat-eq-1-imp-unit*:

fixes $n :: \text{nat}$

assumes $x [\wedge] n = \mathbf{1}$ **and** $n > 0$ **and** [*simp*]: $x \in \text{carrier } G$

shows $x \in \text{Units } G$

<proof>

lemma (*in cring*) *finsum-reindex-bij-betw*:

assumes *bij-betw* $h S T$ $g \in T \rightarrow \text{carrier } R$

shows $\text{finsum } R (\lambda x. g (h x)) S = \text{finsum } R g T$

<proof>

lemma (*in cring*) *finsum-reindex-bij-witness*:

assumes *witness*:

$\bigwedge a. a \in S \implies i (j a) = a$

$\bigwedge a. a \in S \implies j a \in T$

$\bigwedge b. b \in T \implies j (i b) = b$

$\bigwedge b. b \in T \implies i b \in S$

$\bigwedge b. b \in S \implies g b \in \text{carrier } R$

assumes *eq*:

$\bigwedge a. a \in S \implies h (j a) = g a$

shows $\text{finsum } R g S = \text{finsum } R h T$

<proof>

lemma (*in cring*) *binomial*:

```

fixes  $n :: \text{nat}$ 
assumes  $[simp]: x \in \text{carrier } R \ y \in \text{carrier } R$ 
shows  $(x \oplus y) [\wedge] n = (\bigoplus_{i \in \{..n\}}. \text{add-pow } R \ (n \ \text{choose } i) \ (x [\wedge] i \otimes y [\wedge] (n - i)))$ 
<proof>

```

lemma (in *cring*) *binomial-finite-char*:

```

fixes  $p :: \text{nat}$ 
assumes  $[simp]: x \in \text{carrier } R \ y \in \text{carrier } R$  and  $\text{add-pow } R \ p \ \mathbf{1} = \mathbf{0}$  prime  $p$ 
shows  $(x \oplus y) [\wedge] p = x [\wedge] p \oplus y [\wedge] p$ 
<proof>

```

lemma (in *ring-hom-cring*) *hom-add-pow-nat*:

```

 $x \in \text{carrier } R \implies h \ (\text{add-pow } R \ (n :: \text{nat}) \ x) = \text{add-pow } S \ n \ (h \ x)$ 
<proof>

```

end

2 The Lucas–Lehmer test

theory *Lucas-Lehmer*

imports

```

Lucas-Lehmer-Auxiliary
HOL-Algebra.Ring
Probabilistic-Prime-Tests.Jacobi-Symbol
Pell.Pell

```

begin

2.1 General properties of Mersenne numbers and Mersenne primes

We mostly follow the proofs given on Wikipedia [4, 3] in the following sections.

We first show some basic and theorems about Mersenne numbers and Mersenne primes in general, beginning with this: Mersenne primes are the only primes of the form $a^n - 1$ for $n > 1$.

lemma *prime-power-minus-oneD*:

```

fixes  $a \ n :: \text{nat}$ 
assumes prime  $(a \wedge n - 1)$ 
shows  $n = 1 \vee a = 2$ 
<proof>

```

Next, we show that if a prime q divides a Mersenne number $2^p - 1$ with an odd prime exponent p , then q must be of the form $q = 1 + 2kp$ for some $k > 0$.

lemma *prime-dvd-mersenneD*:

```

fixes  $p \ q :: \text{nat}$ 

```

assumes *prime p p ≠ 2 prime q q dvd (2 ^ p - 1)*
shows $[q = 1] \pmod{2 * p}$
 ⟨*proof*⟩

lemma *prime-dvd-mersenneD'*:
fixes *p q :: nat*
assumes *prime p p ≠ 2 prime q q dvd (2 ^ p - 1)*
shows $\exists k > 0. q = 1 + 2 * k * p$
 ⟨*proof*⟩

A Mersenne number is any number of the form $2^p - 1$ for a natural number p . To make things a bit more pleasant, we additionally exclude $2^2 - 1$, i.e. we require $p > 2$. It can be shown that p is then always an odd prime.

locale *mersenne-prime =*
fixes *p M :: nat*
defines $M \equiv 2 ^ p - 1$
assumes *p-gt-2: p > 2 and prime: prime M*
begin

lemma *M-gt-6: M > 6*
 ⟨*proof*⟩

lemma *M-odd: odd M*
 ⟨*proof*⟩

theorem *p-prime: prime p*
 ⟨*proof*⟩

lemma *p-odd: odd p*
 ⟨*proof*⟩

We now first show a few more properties of Mersenne primes regarding congruences and the Legendre symbol.

lemma *M-cong-7-mod-12: [M = 7] (mod 12)*
 ⟨*proof*⟩

lemma *Legendre-3-M: Legendre 3 M = -1*
 ⟨*proof*⟩

lemma *M-cong-7-mod-8: [M = 7] (mod 8)*
 ⟨*proof*⟩

lemma *Legendre-2-M: Legendre 2 M = 1*
 ⟨*proof*⟩

lemma *M-not-dvd-24: ¬M dvd 24*
 ⟨*proof*⟩

end

2.2 The Lucas–Lehmer sequence

We now define the Lucas–Lehmer sequence $a_{n+1} = a_n^2 - 2$. The starting value we will always use is $a_0 = 4$.

primrec *gen-lucas-lehmer-sequence* :: *int* \Rightarrow *nat* \Rightarrow *int* **where**

gen-lucas-lehmer-sequence *a* 0 = *a*
| *gen-lucas-lehmer-sequence* *a* (*Suc* *n*) = *gen-lucas-lehmer-sequence* *a* *n* ^ 2 - 2

lemma *gen-lucas-lehmer-sequence-Suc'*:

gen-lucas-lehmer-sequence *a* (*Suc* *n*) = *gen-lucas-lehmer-sequence* (*a* ^ 2 - 2) *n*
<proof>

lemma *gen-lucas-lehmer-code* [*code*]:

gen-lucas-lehmer-sequence *a* 0 = *a*
gen-lucas-lehmer-sequence *a* (*Suc* *n*) = *gen-lucas-lehmer-sequence* (*a* ^ 2 - 2) *n*
<proof>

For $a_0 = 4$, the recurrence has the closed form $a_{4,n} = \omega^{2^n} + \bar{\omega}^{2^n}$ with $\omega = 2 + \sqrt{3}$ and $\bar{\omega} = 2 - \sqrt{3}$.

lemma *gen-lucas-lehmer-sequence-4-closed-form1*:

real-of-int (*gen-lucas-lehmer-sequence* 4 *n*) = (2 + *sqrt* 3) ^ (2 ^ *n*) + (2 - *sqrt* 3) ^ (2 ^ *n*)
<proof>

lemma *gen-lucas-lehmer-sequence-4-closed-form2*:

gen-lucas-lehmer-sequence 4 *n* = round ((2 + *sqrt* 3) ^ (2 ^ *n*))
<proof>

lemma *gen-lucas-lehmer-sequence-4-closed-form3*:

gen-lucas-lehmer-sequence 4 *n* = [(2 + *sqrt* 3) ^ (2 ^ *n*)]
<proof>

2.3 The ring $\mathbb{Z}[\sqrt{3}]$

To relate this sequence to Mersenne primes, we now first need to define the ring $\mathbb{Z}[\sqrt{3}]$, which is a subring of \mathbb{R} . This ring can be seen as the lattice on \mathbb{R} that is freely generated by 1 and $\sqrt{3}$.

It is, however, more convenient to explicitly describe it as a ring structure over the set $\mathbb{Z} \times \mathbb{Z}$ with a corresponding injective homomorphism $\mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{R}$.

definition *lucas-lehmer-add'* :: *int* \times *int* \Rightarrow *int* \times *int* \Rightarrow *int* \times *int* **where**

lucas-lehmer-add' = ($\lambda(a,b)$ (*c*,*d*). (*a* + *c*, *b* + *d*))

definition *lucas-lehmer-mult'* :: *int* \times *int* \Rightarrow *int* \times *int* \Rightarrow *int* \times *int* **where**

lucas-lehmer-mult' = ($\lambda(a,b)$ (*c*,*d*). (*a* * *c* + 3 * *b* * *d*, *a* * *d* + *b* * *c*))

definition *lucas-lehmer-ring* :: (*int* \times *int*) *ring* **where**

lucas-lehmer-ring =

```

(| carrier = UNIV,
  monoid.mult = lucas-lehmer-mult',
  one = (1, 0),
  ring.zero = (0, 0),
  add = lucas-lehmer-add'|)

```

lemma *carrier-lucas-lehmer-ring* [simp]: *carrier lucas-lehmer-ring = UNIV*
 ⟨proof⟩

lemma *cring-lucas-lehmer-ring* [intro]: *cring (lucas-lehmer-ring)*
 ⟨proof⟩

2.4 The ring $(\mathbb{Z}/m\mathbb{Z})[\sqrt{3}]$

We shall also need the ring $(\mathbb{Z}/m\mathbb{Z})[\sqrt{3}]$, which is obtained from $\mathbb{Z}[\sqrt{3}]$ by reducing each component separately modulo m . This essentially identifies any two points that are a multiple of m apart and then all those that are a multiple of $m\sqrt{3}$ apart.

definition *lucas-lehmer-mult* :: *nat* \Rightarrow *nat* \times *nat* \Rightarrow *nat* \times *nat* \Rightarrow *nat* \times *nat* **where**
lucas-lehmer-mult $m = (\lambda(a,b) (c,d). ((a * c + 3 * b * d) \bmod m, (a * d + b * c) \bmod m))$

definition *lucas-lehmer-add* :: *nat* \Rightarrow *nat* \times *nat* \Rightarrow *nat* \times *nat* \Rightarrow *nat* \times *nat* **where**
lucas-lehmer-add $m = (\lambda(a,b) (c,d). ((a + c) \bmod m, (b + d) \bmod m))$

definition *lucas-lehmer-ring-mod* :: *nat* \Rightarrow (*nat* \times *nat*) **ring where**
lucas-lehmer-ring-mod $m =$
 (| carrier = $\{..<m\} \times \{..<m\}$,
 monoid.mult = *lucas-lehmer-mult* m ,
 one = (1, 0),
 ring.zero = (0, 0),
 add = *lucas-lehmer-add* m)

lemma *lucas-lehmer-add-in-carrier*: $m > 0 \implies \text{lucas-lehmer-add } m \ x \ y \in \{..<m\} \times \{..<m\}$
 ⟨proof⟩

lemma *lucas-lehmer-mult-in-carrier*: $m > 0 \implies \text{lucas-lehmer-mult } m \ x \ y \in \{..<m\} \times \{..<m\}$
 ⟨proof⟩

lemma *lucas-lehmer-add-cong*:
 [fst (*lucas-lehmer-add* $m \ x \ y$) = fst x + fst y] (mod m)
 [snd (*lucas-lehmer-add* $m \ x \ y$) = snd x + snd y] (mod m)
 ⟨proof⟩

lemma *lucas-lehmer-mult-cong*:

$[fst (lucas\text{-}lehmer\text{-}mult\ m\ x\ y) = fst\ x * fst\ y + 3 * snd\ x * snd\ y] (mod\ m)$
 $[snd (lucas\text{-}lehmer\text{-}mult\ m\ x\ y) = fst\ x * snd\ y + snd\ x * fst\ y] (mod\ m)$
 ⟨proof⟩

lemma *lucas-lehmer-add-neutral* [simp]:
assumes $fst\ x < m\ snd\ x < m$
shows $lucas\text{-}lehmer\text{-}add\ m\ (0, 0)\ x = x$
and $lucas\text{-}lehmer\text{-}add\ m\ x\ (0, 0) = x$
 ⟨proof⟩

lemma *lucas-lehmer-mult-neutral* [simp]:
assumes $fst\ x < m\ snd\ x < m$
shows $lucas\text{-}lehmer\text{-}mult\ m\ (Suc\ 0, 0)\ x = x$
and $lucas\text{-}lehmer\text{-}mult\ m\ x\ (Suc\ 0, 0) = x$
 ⟨proof⟩

lemma *lucas-lehmer-add-commute*: $lucas\text{-}lehmer\text{-}add\ m\ x\ y = lucas\text{-}lehmer\text{-}add\ m\ y\ x$
 ⟨proof⟩

lemma *lucas-lehmer-mult-commute*: $lucas\text{-}lehmer\text{-}mult\ m\ x\ y = lucas\text{-}lehmer\text{-}mult\ m\ y\ x$
 ⟨proof⟩

lemma *lucas-lehmer-add-assoc*:
assumes $m: m > 0$
shows $lucas\text{-}lehmer\text{-}add\ m\ x\ (lucas\text{-}lehmer\text{-}add\ m\ y\ z) =$
 $lucas\text{-}lehmer\text{-}add\ m\ (lucas\text{-}lehmer\text{-}add\ m\ x\ y)\ z$
 ⟨proof⟩

lemma *lucas-lehmer-mult-assoc*:
assumes $m: m > 0$
shows $lucas\text{-}lehmer\text{-}mult\ m\ x\ (lucas\text{-}lehmer\text{-}mult\ m\ y\ z) =$
 $lucas\text{-}lehmer\text{-}mult\ m\ (lucas\text{-}lehmer\text{-}mult\ m\ x\ y)\ z$
 ⟨proof⟩

lemma *lucas-lehmer-distrib-right*:
assumes $m: m > 1$
shows $lucas\text{-}lehmer\text{-}mult\ m\ (lucas\text{-}lehmer\text{-}add\ m\ x\ y)\ z =$
 $lucas\text{-}lehmer\text{-}add\ m\ (lucas\text{-}lehmer\text{-}mult\ m\ x\ z)\ (lucas\text{-}lehmer\text{-}mult\ m\ y\ z)$
 ⟨proof⟩

lemma *lucas-lehmer-distrib-left*:
assumes $m > 1$
shows $lucas\text{-}lehmer\text{-}mult\ m\ z\ (lucas\text{-}lehmer\text{-}add\ m\ x\ y) =$
 $lucas\text{-}lehmer\text{-}add\ m\ (lucas\text{-}lehmer\text{-}mult\ m\ z\ x)\ (lucas\text{-}lehmer\text{-}mult\ m\ z\ y)$
 ⟨proof⟩

lemma *cring-lucas-lehmer-ring-mod* [intro]:

assumes $m > 1$
shows $cring$ ($lucas-lehmer-ring-mod\ m$)
 $\langle proof \rangle$

Since 0 is clearly not a unit in the ring and its carrier has size m^2 , the number of units is strictly less than m^2 .

lemma $card-lucas-lehmer-Units$:
assumes $m > 1$
shows $card$ ($Units$ ($lucas-lehmer-ring-mod\ m$)) $< m^2$
 $\langle proof \rangle$

Consider now the case of a prime modulus m : Since $\mathbb{Z}/m\mathbb{Z} = GF(m)$ is a field, any element of $\mathbb{Z}/m\mathbb{Z}$ is a unit in $(\mathbb{Z}/m\mathbb{Z})[\sqrt{3}]$.

lemma $int-in-Units-lucas-lehmer-ring-mod$:
assumes $prime\ p$
assumes $x > 0\ x < p$
shows $(x, 0) \in Units$ ($lucas-lehmer-ring-mod\ p$)
 $\langle proof \rangle$

2.5 $\mathbb{Z}[\sqrt{3}]$ as a subring of \mathbb{R}

We now define the homomorphism from $\mathbb{Z}[\sqrt{3}]$ into the reals:

definition $lucas-lehmer-to-real :: int \times int \Rightarrow real$ **where**
 $lucas-lehmer-to-real = (\lambda(a,b). real-of-int\ a + real-of-int\ b * sqrt\ 3)$

context
begin

interpretation $cring\ lucas-lehmer-ring$ $\langle proof \rangle$

lemma $minus-lucas-lehmer-ring$: $\ominus_{lucas-lehmer-ring}\ x = (case\ x\ of\ (a, b) \Rightarrow (-a, -b))$
 $\langle proof \rangle$

lemma $lucas-lehmer-to-real-simps1$:
 $lucas-lehmer-to-real\ (a, b) = of-int\ a + of-int\ b * sqrt\ 3$
 $lucas-lehmer-to-real\ (x \oplus_{lucas-lehmer-ring}\ y) = lucas-lehmer-to-real\ x + lucas-lehmer-to-real\ y$
 $lucas-lehmer-to-real\ (x \otimes_{lucas-lehmer-ring}\ y) = lucas-lehmer-to-real\ x * lucas-lehmer-to-real\ y$
 $lucas-lehmer-to-real\ (\ominus_{lucas-lehmer-ring}\ x) = -lucas-lehmer-to-real\ x$
 $lucas-lehmer-to-real\ (\mathbf{0}_{lucas-lehmer-ring}) = 0$
 $lucas-lehmer-to-real\ (\mathbf{1}_{lucas-lehmer-ring}) = 1$
 $\langle proof \rangle$

lemma $lucas-lehmer-to-add-pow-nat$:
 $lucas-lehmer-to-real\ ([n] \cdot_{lucas-lehmer-ring}\ x) = of-nat\ n * lucas-lehmer-to-real\ x$
 $\langle proof \rangle$

lemma *lucas-lehmer-to-add-pow-int*:

lucas-lehmer-to-real ($[n] \cdot \text{lucas-lehmer-ring } x$) = *of-int* $n * \text{lucas-lehmer-to-real } x$
<proof>

lemma *lucas-lehmer-to-real-power*:

lucas-lehmer-to-real ($x \widehat{[]} \text{lucas-lehmer-ring } (n :: \text{nat})) = \text{lucas-lehmer-to-real } x \widehat{^n}$
<proof>

lemmas *lucas-lehmer-to-real-simps =*

lucas-lehmer-to-real-simps1 lucas-lehmer-to-real-power
lucas-lehmer-to-add-pow-nat lucas-lehmer-to-add-pow-int

end

lemma *lucas-lehmer-to-real-inj*: *inj lucas-lehmer-to-real*

<proof>

2.6 The canonical homomorphism $\mathbb{Z}[\sqrt{3}] \rightarrow (\mathbb{Z}/m\mathbb{Z})[\sqrt{3}]$

Next, we show that reduction modulo m is indeed a homomorphism.

definition *lucas-lehmer-hom* :: $\text{nat} \Rightarrow (\text{int} \times \text{int}) \Rightarrow (\text{nat} \times \text{nat})$ **where**

lucas-lehmer-hom $m = (\lambda(x,y). (\text{nat } (x \bmod m), \text{nat } (y \bmod m)))$

lemma *lucas-lehmer-hom-cong*:

$[\text{fst } x = \text{fst } y] (\bmod \text{int } m) \Longrightarrow [\text{snd } x = \text{snd } y] (\bmod \text{int } m) \Longrightarrow$
lucas-lehmer-hom m $x = \text{lucas-lehmer-hom } m$ y
<proof>

lemma *lucas-lehmer-hom-cong'*:

$[a = b] (\bmod \text{int } m) \Longrightarrow [c = d] (\bmod \text{int } m) \Longrightarrow$
lucas-lehmer-hom m $(a, c) = \text{lucas-lehmer-hom } m$ (b, d)
<proof>

context

fixes $m :: \text{nat}$

assumes $m: m > 1$

begin

lemma *lucas-lehmer-hom-in-carrier*: *lucas-lehmer-hom* m $x \in \{..<m\} \times \{..<m\}$

<proof>

lemma *lucas-lehmer-hom-add*:

lucas-lehmer-hom m (*lucas-lehmer-add'* x y) =
lucas-lehmer-add m (*lucas-lehmer-hom* m x) (*lucas-lehmer-hom* m y)
<proof>

lemma *lucas-lehmer-hom-mult*:

$lucas\text{-}lehmer\text{-}hom\ m\ (lucas\text{-}lehmer\text{-}mult'\ x\ y) =$
 $lucas\text{-}lehmer\text{-}mult\ m\ (lucas\text{-}lehmer\text{-}hom\ m\ x)\ (lucas\text{-}lehmer\text{-}hom\ m\ y)$
 <proof>

lemma *lucas-lehmer-hom-1* [*simp*]: $lucas\text{-}lehmer\text{-}hom\ m\ (1, 0) = (1, 0)$
 <proof>

lemma *ring-hom-lucas-lehmer-hom*:
 $lucas\text{-}lehmer\text{-}hom\ m \in ring\text{-}hom\ lucas\text{-}lehmer\text{-}ring\ (lucas\text{-}lehmer\text{-}ring\text{-}mod\ m)$
 <proof>

end

2.7 Correctness of the Lucas–Lehmer test

In this section, we will prove that the Lucas–Lehmer test is both a necessary and sufficient condition for the primality of a Mersenne number of the form $2^p - 1$ for an odd prime p . The proof that shall be given here is rather explicit and heavily draws from the Wikipedia article on the Lucas–Lehmer test [3].

A shorter and more high-level proof of a more general statement can be obtained using more theory on finite fields (in particular the field $\text{GF}(q^2)$) (cf. e. g. Rödseth [2]).

definition *lucas-lehmer-test where*
 $lucas\text{-}lehmer\text{-}test\ p = (p > 2 \wedge$
 $(2 \wedge p - 1)\ dvd\ gen\text{-}lucas\text{-}lehmer\text{-}sequence\ 4\ (p - 2))$

We can now prove that any Mersenne number $2^p - 1$ for p prime that passes the Lucas–Lehmer test is prime. We follow the simple argument given by Bruce [1], which is also given on Wikipedia [3].

theorem *lucas-lehmer-sufficient*:
assumes *prime p odd p*
assumes $(2 \wedge p - 1)\ dvd\ gen\text{-}lucas\text{-}lehmer\text{-}sequence\ 4\ (p - 2)$
shows $prime\ (2 \wedge p - 1 :: nat)$
 <proof>

Next, we show that any Mersenne prime passes the Lucas–Lehmer test. We again follow the rather explicit proof outlined on Wikipedia [3], which is a simplified (but less general and less abstract) version of the proof by Rödseth [2].

theorem (**in** *mersenne-prime*) *lucas-lehmer-necessary*:
 $(2 \wedge p - 1)\ dvd\ gen\text{-}lucas\text{-}lehmer\text{-}sequence\ 4\ (p - 2)$
 <proof>

corollary *lucas-lehmer-correct*:
 $prime\ (2 \wedge p - 1 :: nat) \longleftrightarrow$

$prime\ p \wedge (p = 2 \vee (2 \wedge p - 1) \text{ dvd } gen\text{-lucas-lehmer-sequence}\ 4\ (p - 2))$
 ⟨proof⟩

corollary *lucas-lehmer-correct'*:

$prime\ (2 \wedge p - 1 :: nat) \longleftrightarrow prime\ p \wedge (p = 2 \vee lucas\text{-lehmer-test}\ p)$
 ⟨proof⟩

2.8 A first executable version Lucas–Lehmer test

The following is an implementation of the Lucas–Lehmer test using modular arithmetic on the integers. This is not the most efficient implementation – the modular arithmetic can be replaced by much cheaper bitwise operations, and we will do that in the next section.

primrec *gen-lucas-lehmer-sequence'* :: $int \Rightarrow int \Rightarrow nat \Rightarrow int$ **where**

gen-lucas-lehmer-sequence' $m\ a\ 0 = a$
 | *gen-lucas-lehmer-sequence'* $m\ a\ (Suc\ n) = gen\text{-lucas-lehmer-sequence}'\ m\ ((a \wedge 2 - 2) \bmod m)\ n$

lemma *gen-lucas-lehmer-sequence'-Suc'*:

gen-lucas-lehmer-sequence' $m\ a\ (Suc\ n) = (gen\text{-lucas-lehmer-sequence}'\ m\ a\ n \wedge 2 - 2) \bmod m$
 ⟨proof⟩

lemma *gen-lucas-lehmer-sequence'-correct*:

assumes $a \in \{0..<m\}$
shows $gen\text{-lucas-lehmer-sequence}'\ m\ a\ n = gen\text{-lucas-lehmer-sequence}\ a\ n \bmod m$
 ⟨proof⟩

lemma *lucas-lehmer-test-code-arithmetic* [code]:

lucas-lehmer-test $p = (p > 2 \wedge gen\text{-lucas-lehmer-sequence}'\ (2 \wedge p - 1)\ 4\ (p - 2) = 0)$
 ⟨proof⟩

lemma *mersenne-prime-iff*: $mersenne\text{-prime}\ p \longleftrightarrow p > 2 \wedge prime\ (2 \wedge p - 1 :: nat)$

⟨proof⟩

lemma *mersenne-prime-code* [code]:

$mersenne\text{-prime}\ p \longleftrightarrow prime\ p \wedge lucas\text{-lehmer-test}\ p$
 ⟨proof⟩

end

3 Efficient code for testing Mersenne primes

theory *Lucas-Lehmer-Code*

imports

Lucas-Lehmer
HOL-Library.Code-Target-Numeral
HOL-Library.Code-Bit-Shifts-for-Arithmetic
begin

3.1 Efficient computation of remainders modulo a Mersenne number

We have $k = k \bmod 2^n + k \operatorname{div} 2^n \pmod{(2^n - 1)}$, and $k \bmod 2^n = k \&t(2^n - 1)$ and $k \operatorname{div} 2^n = k \gg n$. Therefore, we can reduce k modulo $2^n - 1$ using only bitwise operations, addition, and bit shifts.

lemma *cong-mersenne-number-int*:
fixes $k :: \text{int}$
shows $[k \bmod 2^n + k \operatorname{div} 2^n = k] \pmod{(2^n - 1)}$
 $\langle \text{proof} \rangle$

We encapsulate a single reduction step in the following operation. Note, however, that the result is not, in general, the same as $k \bmod (2^n - 1)$. Multiple reductions might be required in order to reduce it below 2^n , and a multiple of $2^n - 1$ can be reduced to $2^n - 1$, which is invariant to further reduction steps.

definition *mersenne-mod* $:: \text{int} \Rightarrow \text{nat} \Rightarrow \text{int}$ **where**
mersenne-mod $k\ n = k \bmod 2^n + k \operatorname{div} 2^n$

lemma *mersenne-mod-code* [*code*]:
mersenne-mod $k\ n = \text{take-bit } n\ k + \text{drop-bit } n\ k$
 $\langle \text{proof} \rangle$

lemma *cong-mersenne-mod*: $[\text{mersenne-mod } k\ n = k] \pmod{(2^n - 1)}$
 $\langle \text{proof} \rangle$

lemma *mersenne-mod-nonneg* [*simp*]: $k \geq 0 \implies \text{mersenne-mod } k\ n \geq 0$
 $\langle \text{proof} \rangle$

lemma *mersenne-mod-less*:
assumes $k \leq 2^m\ m \geq n$
shows $\text{mersenne-mod } k\ n < 2^n + 2^{(m - n)}$
 $\langle \text{proof} \rangle$

lemma *mersenne-mod-less'*:
assumes $k \leq 5 * 2^n$
shows $\text{mersenne-mod } k\ n < 2^n + 5$
 $\langle \text{proof} \rangle$

It turns out that for our use case, a single reduction is not enough to reduce the number in question enough (or at least I was unable to prove that it is). We therefore perform two reduction steps, which is enough to guarantee

that our numbers are below $2^n + 4$ before and after every step in the Lucas–Lehmer sequence.

Whether one or two reductions are performed is not very important anyway, since the dominant step is the squaring anyway.

definition *mersenne-mod2* :: *int* ⇒ *nat* ⇒ *int* **where**
mersenne-mod2 *k n* = *mersenne-mod* (*mersenne-mod* *k n*) *n*

lemma *cong-mersenne-mod2*: [*mersenne-mod2* *k n* = *k*] (mod ($2^n - 1$))
 ⟨*proof*⟩

lemma *mersenne-mod2-nonneg* [*simp*]: $k \geq 0 \implies \text{mersenne-mod2 } k \ n \geq 0$
 ⟨*proof*⟩

lemma *mersenne-mod2-less*:
assumes $n > 2$ **and** $k \leq 2^{2 * n + 2}$
shows *mersenne-mod2* *k n* < $2^n + 5$
 ⟨*proof*⟩

Since we subtract 2 at one point, the intermediate results can become negative. This is not a problem since our reduction modulo $2^p - 1$ happens to make them positive again immediately.

lemma *mersenne-mod-nonneg-strong*:
 ⟨*mersenne-mod* *a p* ≥ 0⟩ **if** ⟨ $-(2^p) + 1 < a$ ⟩
 ⟨*proof*⟩

lemma *mersenne-mod2-nonneg-strong*:
assumes $a > -(2^p) + 1$
shows *mersenne-mod2* *a p* ≥ 0
 ⟨*proof*⟩

3.2 Efficient code for the Lucas–Lehmer sequence

primrec *gen-lucas-lehmer-sequence''* :: *nat* ⇒ *int* ⇒ *nat* ⇒ *int* **where**
gen-lucas-lehmer-sequence'' *p a 0* = *a*
 | *gen-lucas-lehmer-sequence''* *p a* (*Suc n*) =
gen-lucas-lehmer-sequence'' *p* (*mersenne-mod2* ($a^2 - 2$) *p*) *n*

lemma *gen-lucas-lehmer-sequence''-correct*:
assumes [$a = a'$] (mod ($2^p - 1$))
shows [*gen-lucas-lehmer-sequence''* *p a n* = *gen-lucas-lehmer-sequence* *a' n*]
 (mod ($2^p - 1$))
 ⟨*proof*⟩

lemma *gen-lucas-lehmer-sequence''-bounds*:
assumes $a \geq 0$ $a < 2^p + 5$ $p > 2$
shows *gen-lucas-lehmer-sequence''* *p a n* ∈ {0.. $2^p + 5$ }
 ⟨*proof*⟩

3.3 Code for the Lucas–Lehmer test

lemma *lucas-lehmer-test-code* [code]:

```
lucas-lehmer-test p  $\longleftrightarrow$   
  (2 < p  $\wedge$  (let x = gen-lucas-lehmer-sequence'' p 4 (p - 2) in x = 0  $\vee$  x =  
(push-bit p 1) - 1))  
  <proof>
```

3.4 Examples

Note that for some reason, the clever bit-arithmetic version of the Lucas–Lehmer test is actually much slower than the one using integer arithmetic when using PolyML, and even more so when using the built-in evaluator in Isabelle (which also uses PolyML with a slightly different setup).

I do not quite know why this is the case, but it is likely because of inefficient implementations of bit arithmetic operations in PolyML and/or the code generator setup for it.

When running with GHC, the bit-arithmetic version is *much* faster.

value *filter mersenne-prime* [0..<100]

lemma *prime* (2 $^$ 521 - 1 :: nat)
 <proof>

lemma *prime* (2 $^$ 4253 - 1 :: nat)
 <proof>

end

References

- [1] J. W. Bruce. A really trivial proof of the Lucas-Lehmer test. *The American Mathematical Monthly*, 100(4):370–371, 1993.
- [2] Ö. J. Rödseth. A note on primality tests for $n = h \cdot 2^n - 1$. *BIT Numerical Mathematics*, 34(3):451–454, Sep 1994.
- [3] Wikipedia contributors. Lucas–Lehmer primality test — Wikipedia, the free encyclopedia, 2020. [Online; accessed 17 Jan 2020].
- [4] Wikipedia contributors. Mersenne prime — Wikipedia, the free encyclopedia, 2020. [Online; accessed 17 Jan 2020].