Mersenne primes and the Lucas–Lehmer test

Manuel Eberl

March 17, 2025

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 $\mathbb{Z}[\sqrt{3}] \to (\mathbb{Z}/m\mathbb{Z})[\sqrt{3}]$ 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 14
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*: $[a = b]\ (\mathit{mod}\ n) \Longrightarrow a\ \mathit{mod}\ n = b\ \mathit{mod}\ n$
⟨*proof*⟩

lemma *eval-coprime*:
$(b :: {'a} :: \mathit{euclidean\text{-}semiring\text{-}gcd}) \neq 0 \Longrightarrow \mathit{coprime}\ a\ b \longleftrightarrow \mathit{coprime}\ b\ (a\ \mathit{mod}\ b)$
⟨*proof*⟩

lemma *two-power-odd-mod-12*:
assumes $\mathit{odd}\ n\ n > 1$
shows $[2\ \hat{}\ n = 8]\ (\mathit{mod}\ (12 :: \mathit{nat}))$
⟨*proof*⟩

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

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

lemma *supplement2-Legendre′*:
assumes $\mathit{prime}\ p\ p \neq 2$
shows $\mathit{Legendre}\ 2\ p = (\mathit{if}\ p\ \mathit{mod}\ 8 = 1 \lor p\ \mathit{mod}\ 8 = 7\ \mathit{then}\ 1\ \mathit{else}\ {-1})$
⟨*proof*⟩

lemma *little-Fermat-nat*:
fixes $a :: \mathit{nat}$
assumes $\mathit{prime}\ p\ \neg p\ \mathit{dvd}\ a$
shows $[a\ \hat{}\ p = a]\ (\mathit{mod}\ p)$
⟨*proof*⟩

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

⟨*proof*⟩

lemma *prime-dvd-choose*:
  assumes $0 < k$ $k < p$ *prime p*
  shows *p dvd* (*p choose k*)
⟨*proof*⟩

lemma *prime-natD*:
  assumes *prime* (*p* :: *nat*) *a dvd p*
  shows $a = 1 \lor a = p$
  ⟨*proof*⟩

  lemma *not-prime-imp-ex-prod-nat*:
  assumes $m > 1$ $\neg$ *prime* (*m*::*nat*)
  shows $\exists n\ k.\ m = n * k \land 1 < n \land n < m \land 1 < k \land k < m$
⟨*proof*⟩

1.2 Auxiliary algebraic material

lemma (**in** *group*) *ord-eqI-prime-factors*:
  assumes $\bigwedge p.\ p \in$ *prime-factors* $n \Longrightarrow x\ [\urcorner]\ (n\ div\ p) \neq \mathbf{1}$ **and** $x\ [\urcorner]\ n = \mathbf{1}$
  assumes $x \in$ *carrier* G $n > 0$
  shows *group.ord* $G\ x = n$
⟨*proof*⟩

lemma (**in** *monoid*) *pow-nat-eq-1-imp-unit*:
  fixes *n* :: *nat*
  assumes $x\ [\urcorner]\ n = \mathbf{1}$ **and** $n > 0$ **and** [*simp*]: $x \in$ *carrier G*
  shows $x \in$ *Units G*
⟨*proof*⟩

lemma (**in** *cring*) *finsum-reindex-bij-betw*:
  assumes *bij-betw h S T* $g \in T \to$ *carrier R*
  shows *finsum R* $(\lambda x.\ g\ (h\ x))$ $S =$ *finsum R g T*
  ⟨*proof*⟩

lemma (**in** *cring*) *finsum-reindex-bij-witness*:
  assumes *witness*:
    $\bigwedge a.\ a \in S \Longrightarrow i\ (j\ a) = a$
    $\bigwedge a.\ a \in S \Longrightarrow j\ a \in T$
    $\bigwedge b.\ b \in T \Longrightarrow j\ (i\ b) = b$
    $\bigwedge b.\ b \in T \Longrightarrow i\ b \in S$
    $\bigwedge b.\ b \in S \Longrightarrow g\ b \in$ *carrier R*
  assumes *eq*:
    $\bigwedge a.\ a \in S \Longrightarrow h\ (j\ a) = g\ a$
  shows *finsum R g S* = *finsum R h T*
⟨*proof*⟩

lemma (**in** *cring*) *binomial*:

fixes *n* :: *nat*
assumes [*simp*]: *x* ∈ *carrier R y* ∈ *carrier R*
shows (*x* ⊕ *y*) [^] *n* = (⨁ *i*∈{..*n*}. *add-pow R* (*n choose i*) (*x* [^] *i* ⊗ *y* [^] (*n* − *i*)))
⟨*proof*⟩

lemma (**in** *cring*) *binomial-finite-char*:
fixes *p* :: *nat*
assumes [*simp*]: *x* ∈ *carrier R y* ∈ *carrier R* **and** *add-pow R p* **1** = **0** *prime p*
shows (*x* ⊕ *y*) [^] *p* = *x* [^] *p* ⊕ *y* [^] *p*
⟨*proof*⟩

lemma (**in** *ring-hom-cring*) *hom-add-pow-nat*:
x ∈ *carrier R* ⟹ *h* (*add-pow R* (*n*::*nat*) *x*) = *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* :: *nat*
assumes *prime* (*a* ^ *n* − *1*)
shows *n* = *1* ∨ *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* :: *nat*

  assumes *prime p p ≠ 2 prime q q dvd (2 ^ p − 1)*
  shows *[q = 1] (mod (2 ∗ p))*
⟨*proof*⟩

lemma *prime-dvd-mersenneD′*:
  fixes *p q :: nat*
  assumes *prime p p ≠ 2 prime q q dvd (2 ^ p − 1)*
  shows *∃ 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 ≡ 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* ⇒ *nat* ⇒ *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*⟩

lemmas *gen-lucas-lehmer-code* [*code*] =
  gen-lucas-lehmer-sequence.simps(*1*) *gen-lucas-lehmer-sequence-Suc'*

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 $\mathbb{Z} \times \mathbb{Z} \to \mathbb{R}$.

definition *lucas-lehmer-add'* :: *int* × *int* ⇒ *int* × *int* ⇒ *int* × *int* **where**
  lucas-lehmer-add' = (λ(*a*,*b*) (*c*,*d*). (*a* + *c*, *b* + *d*))

definition *lucas-lehmer-mult'* :: *int* × *int* ⇒ *int* × *int* ⇒ *int* × *int* **where**
  lucas-lehmer-mult' = (λ(*a*,*b*) (*c*,*d*). (*a* ∗ *c* + *3* ∗ *b* ∗ *d*, *a* ∗ *d* + *b* ∗ *c*))

definition *lucas-lehmer-ring* :: (*int* × *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* ⇒ *nat* × *nat* ⇒ *nat* × *nat* ⇒ *nat* × *nat* **where**
lucas-lehmer-mult m = (λ(*a*,*b*) (*c*,*d*). ((*a* ∗ *c* + *3* ∗ *b* ∗ *d*) *mod m*, (*a* ∗ *d* + *b* ∗ *c*) *mod m*))

definition *lucas-lehmer-add* :: *nat* ⇒ *nat* × *nat* ⇒ *nat* × *nat* ⇒ *nat* × *nat* **where**
lucas-lehmer-add m = (λ(*a*,*b*) (*c*,*d*). ((*a* + *c*) *mod m*, (*b* + *d*) *mod m*))

definition *lucas-lehmer-ring-mod* :: *nat* ⇒ (*nat* × *nat*) *ring* **where**
lucas-lehmer-ring-mod m =
⦇*carrier* = {..<*m*} × {..<*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* ⟹ *lucas-lehmer-add m x y* ∈ {..<*m*} × {..<*m*}
⟨*proof*⟩

lemma *lucas-lehmer-mult-in-carrier*: *m* > *0* ⟹ *lucas-lehmer-mult m x y* ∈ {..<*m*} × {..<*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-lehmer-mult m x y*) = *fst x* ∗ *fst y* + *3* ∗ *snd x* ∗ *snd y*] (*mod m*)
[*snd* (*lucas-lehmer-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-lehmer-add m* (*0*, *0*) *x* = *x*
and *lucas-lehmer-add m x* (*0*, *0*) = *x*
⟨*proof*⟩

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

lemma *lucas-lehmer-add-commute*: *lucas-lehmer-add m x y* = *lucas-lehmer-add m y x*
⟨*proof*⟩

lemma *lucas-lehmer-mult-commute*: *lucas-lehmer-mult m x y* = *lucas-lehmer-mult m y x*
⟨*proof*⟩

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

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

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

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

lemma *cring-lucas-lehmer-ring-mod* [*intro*]:
assumes $m > 1$
shows *cring* (*lucas-lehmer-ring-mod m*)

⟨*proof*⟩

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*
⟨*proof*⟩

Consider now the case of a prime modulus m: Since $\mathbb{Z}/m\mathbb{Z} = \mathrm{GF}(m)$ 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) ∈ Units (lucas-lehmer-ring-mod p)*
⟨*proof*⟩

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 × int ⇒ real* **where**
  lucas-lehmer-to-real = (λ(a,b). real-of-int a + real-of-int b ∗ sqrt 3)

context
begin

interpretation *cring lucas-lehmer-ring* ⟨*proof*⟩

lemma *minus-lucas-lehmer-ring*: $\ominus_{lucas\text{-}lehmer\text{-}ring}$ *x = (case x of (a, b) ⇒ (−a, −b))*
  ⟨*proof*⟩

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 $\oplus_{lucas\text{-}lehmer\text{-}ring}$ *y) =*
     lucas-lehmer-to-real x + lucas-lehmer-to-real y
    lucas-lehmer-to-real (x $\otimes_{lucas\text{-}lehmer\text{-}ring}$ *y) =*
     lucas-lehmer-to-real x ∗ lucas-lehmer-to-real y
    lucas-lehmer-to-real ($\ominus_{lucas\text{-}lehmer\text{-}ring}$ *x) = −lucas-lehmer-to-real x*
    lucas-lehmer-to-real ($\mathbf{0}_{lucas\text{-}lehmer\text{-}ring}$*) = 0*
    lucas-lehmer-to-real ($\mathbf{1}_{lucas\text{-}lehmer\text{-}ring}$*) = 1*
  ⟨*proof*⟩

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

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

lucas-lehmer-to-real ([*n*] $\cdot_{\mathit{lucas\text{-}lehmer\text{-}ring}}$ *x*) = *of-int n* ∗ *lucas-lehmer-to-real x*
⟨*proof*⟩

lemma *lucas-lehmer-to-real-power*:
lucas-lehmer-to-real (*x* $[\char`\^]_{\mathit{lucas\text{-}lehmer\text{-}ring}}$ (*n* :: *nat*)) = *lucas-lehmer-to-real x* ^ *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 $\mathbb{Z}[\sqrt{3}] \to (\mathbb{Z}/m\mathbb{Z})[\sqrt{3}]$

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

definition *lucas-lehmer-hom* :: *nat* ⇒ (*int* × *int*) ⇒ (*nat* × *nat*) **where**
lucas-lehmer-hom m = (λ(*x*,*y*). (*nat* (*x mod m*), *nat* (*y mod m*)))

lemma *lucas-lehmer-hom-cong*:
[*fst x* = *fst y*] (*mod int m*) ⟹ [*snd x* = *snd y*] (*mod int m*) ⟹
lucas-lehmer-hom m x = *lucas-lehmer-hom m y*
⟨*proof*⟩

lemma *lucas-lehmer-hom-cong′*:
[*a* = *b*] (*mod int m*) ⟹ [*c* = *d*] (*mod int m*) ⟹
lucas-lehmer-hom m (*a*, *c*) = *lucas-lehmer-hom m* (*b*, *d*)
⟨*proof*⟩

context
fixes *m* :: *nat*
assumes *m*: *m* > *1*
begin

lemma *lucas-lehmer-hom-in-carrier*: *lucas-lehmer-hom m x* ∈ {..<*m*} × {..<*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-lehmer-hom m (*lucas-lehmer-mult′ x y*) =
lucas-lehmer-mult m (*lucas-lehmer-hom m x*) (*lucas-lehmer-hom m y*)

⟨*proof*⟩

lemma *lucas-lehmer-hom-1* [*simp*]: *lucas-lehmer-hom m* (*1*, *0*) = (*1*, *0*)
⟨*proof*⟩

lemma *ring-hom-lucas-lehmer-hom*:
lucas-lehmer-hom m ∈ *ring-hom lucas-lehmer-ring* (*lucas-lehmer-ring-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 $\mathrm{GF}(q^2)$) (cf. e. g. Rödseth [2]).

definition *lucas-lehmer-test* **where**
lucas-lehmer-test p = (*p* > *2* ∧
(*2* ^ *p* − *1*) *dvd gen-lucas-lehmer-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* ^ *p* − *1*) *dvd gen-lucas-lehmer-sequence 4* (*p* − *2*)
shows *prime* (*2* ^ *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* ^ *p* − *1*) *dvd gen-lucas-lehmer-sequence 4* (*p* − *2*)
⟨*proof*⟩

corollary *lucas-lehmer-correct*:
prime (*2* ^ *p* − *1* :: *nat*) ⟷
prime p ∧ (*p* = *2* ∨ (*2* ^ *p* − *1*) *dvd gen-lucas-lehmer-sequence 4* (*p* − *2*))
⟨*proof*⟩

corollary *lucas-lehmer-correct′*:
prime (*2* ^ *p* − *1* :: *nat*) ⟷ *prime p* ∧ (*p* = *2* ∨ *lucas-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* ⇒ *int* ⇒ *nat* ⇒ *int* **where**
gen-lucas-lehmer-sequence′ m a 0 = *a*
| *gen-lucas-lehmer-sequence′ m a* (*Suc n*) = *gen-lucas-lehmer-sequence′ m* ((*a* ^ *2* − *2*) *mod m*) *n*

lemma *gen-lucas-lehmer-sequence′-Suc′*:
gen-lucas-lehmer-sequence′ m a (*Suc n*) = (*gen-lucas-lehmer-sequence′ m a n* ^ *2* − *2*) *mod m*
⟨*proof*⟩

lemma *gen-lucas-lehmer-sequence′-correct*:
assumes *a* ∈ {*0..*<*m*}
shows *gen-lucas-lehmer-sequence′ m a n* = *gen-lucas-lehmer-sequence a n mod m*
⟨*proof*⟩

lemma *lucas-lehmer-test-code-arithmetic* [*code*]:
lucas-lehmer-test p = (*p* > *2* ∧
gen-lucas-lehmer-sequence′ (*2* ^ *p* − *1*) *4* (*p* − *2*) = *0*)
⟨*proof*⟩

lemma *mersenne-prime-iff*: *mersenne-prime p* ⟷ *p* > *2* ∧ *prime* (*2* ^ *p* − *1* :: *nat*)
⟨*proof*⟩

lemma *mersenne-prime-code* [*code*]:
mersenne-prime p ⟷ *prime p* ∧ *lucas-lehmer-test p*
⟨*proof*⟩

end

3 Efficient code for testing Mersenne primes

theory *Lucas-Lehmer-Code*
imports
Lucas-Lehmer
HOL−Library.Code-Target-Numeral

Native-Word.Code-Target-Int-Bit
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 $k \bmod 2^n = k \mathbin{\&} (2^n - 1)$ 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* :: *int*
shows [*k mod 2* ^ *n* + *k div 2* ^ *n* = *k*] (*mod* (*2* ^ *n* − *1*))
⟨*proof*⟩

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* :: *int* ⇒ *nat* ⇒ *int* **where**
mersenne-mod k n = *k mod 2* ^ *n* + *k div 2* ^ *n*

lemma *mersenne-mod-code* [*code*]:
mersenne-mod k n = *take-bit n k* + *drop-bit n k*
⟨*proof*⟩

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

lemma *mersenne-mod-nonneg* [*simp*]: *k* ≥ *0* ⟹ *mersenne-mod k n* ≥ *0*
⟨*proof*⟩

lemma *mersenne-mod-less*:
assumes *k* ≤ *2* ^ *m m* ≥ *n*
shows *mersenne-mod k n* < *2* ^ *n* + *2* ^ (*m* − *n*)
⟨*proof*⟩

lemma *mersenne-mod-less'*:
assumes *k* ≤ *5* ∗ *2* ^ *n*
shows *mersenne-mod k n* < *2* ^ *n* + *5*
⟨*proof*⟩

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 $\mathit{mersenne\text{-}mod2} :: \mathit{int} \Rightarrow \mathit{nat} \Rightarrow \mathit{int}$ **where**
$\mathit{mersenne\text{-}mod2}\ k\ n = \mathit{mersenne\text{-}mod}\ (\mathit{mersenne\text{-}mod}\ k\ n)\ n$

lemma *cong-mersenne-mod2*: $[\mathit{mersenne\text{-}mod2}\ k\ n = k]\ (\mathit{mod}\ (2\ \hat{}\ n - 1))$
$\langle \mathit{proof} \rangle$

lemma *mersenne-mod2-nonneg* [*simp*]: $k \geq 0 \Longrightarrow \mathit{mersenne\text{-}mod2}\ k\ n \geq 0$
$\langle \mathit{proof} \rangle$

lemma *mersenne-mod2-less*:
assumes $n > 2$ **and** $k \leq 2\ \hat{}\ (2 * n + 2)$
shows $\mathit{mersenne\text{-}mod2}\ k\ n < 2\ \hat{}\ n + 5$
$\langle \mathit{proof} \rangle$

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*:
‹$\mathit{mersenne\text{-}mod}\ a\ p \geq 0$› **if** ‹$-\ (2\ \hat{}\ p) + 1 < a$›
$\langle \mathit{proof} \rangle$

lemma *mersenne-mod2-nonneg-strong*:
assumes $a > -(2\ \hat{}\ p) + 1$
shows $\mathit{mersenne\text{-}mod2}\ a\ p \geq 0$
$\langle \mathit{proof} \rangle$

3.2 Efficient code for the Lucas–Lehmer sequence

primrec $\mathit{gen\text{-}lucas\text{-}lehmer\text{-}sequence}'' :: \mathit{nat} \Rightarrow \mathit{int} \Rightarrow \mathit{nat} \Rightarrow \mathit{int}$ **where**
$\mathit{gen\text{-}lucas\text{-}lehmer\text{-}sequence}''\ p\ a\ 0 = a$
$|\ \mathit{gen\text{-}lucas\text{-}lehmer\text{-}sequence}''\ p\ a\ (\mathit{Suc}\ n) =$
$\mathit{gen\text{-}lucas\text{-}lehmer\text{-}sequence}''\ p\ (\mathit{mersenne\text{-}mod2}\ (a\ \hat{}\ 2 - 2)\ p)\ n$

lemma *gen-lucas-lehmer-sequence′′-correct*:
assumes $[a = a']\ (\mathit{mod}\ (2\ \hat{}\ p - 1))$
shows $[\mathit{gen\text{-}lucas\text{-}lehmer\text{-}sequence}''\ p\ a\ n = \mathit{gen\text{-}lucas\text{-}lehmer\text{-}sequence}\ a'\ n]\ (\mathit{mod}\ (2\ \hat{}\ p - 1))$
$\langle \mathit{proof} \rangle$

lemma *gen-lucas-lehmer-sequence′′-bounds*:
assumes $a \geq 0\ a < 2\ \hat{}\ p + 5\ p > 2$
shows $\mathit{gen\text{-}lucas\text{-}lehmer\text{-}sequence}''\ p\ a\ n \in \{0..{<}2\ \hat{}\ p + 5\}$
$\langle \mathit{proof} \rangle$

3.3 Code for the Lucas–Lehmer test

lemmas [*code del*] = *lucas-lehmer-test-code-arithmetic*

lemma *lucas-lehmer-test-code* [*code*]:
lucas-lehmer-test p =
(*2* < *p* ∧ (*let x* = *gen-lucas-lehmer-sequence″ p 4* (*p* − *2*) *in x* = *0* ∨ *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].