

The Transcendence of π

Manuel Eberl

May 26, 2024

Abstract

This entry shows the transcendence of π based on the classic proof using the fundamental theorem of symmetric polynomials first given by von Lindemann in 1882, but the mostly formalisation follows the version by Niven [3]. The proof reuses much of the machinery developed in the AFP entry on the transcendence of e .

Contents

1	The Transcendence of π	2
----------	--	----------

1 The Transcendence of π

theory *Pi-Transcendental*

imports

E-Transcendental.E-Transcendental
Symmetric-Polynomials.Symmetric-Polynomials
HOL-Real-Asymp.Real-Asymp

begin

lemma *ring-homomorphism-to-poly* [intro]: *ring-homomorphism* ($\lambda i. [i:]$)
by *standard auto*

lemma (**in** *ring-closed*) *coeff-power-closed*:
 $(\bigwedge m. \text{coeff } p \ m \in A) \implies \text{coeff } (p \wedge n) \ m \in A$
by (*induction n arbitrary: m*)
(auto simp: mpoly-coeff-1 coeff-mpoly-times intro!: prod-fun-closed)

lemma (**in** *ring-closed*) *coeff-prod-closed*:
 $(\bigwedge x \ m. x \in X \implies \text{coeff } (f \ x) \ m \in A) \implies \text{coeff } (\text{prod } f \ X) \ m \in A$
by (*induction X arbitrary: m rule: infinite-finite-induct*)
(auto simp: mpoly-coeff-1 coeff-mpoly-times intro!: prod-fun-closed)

lemma *map-of-rat-of-int-poly* [simp]: *map-poly of-rat (of-int-poly p) = of-int-poly p*
by (*intro poly-eqI (auto simp: coeff-map-poly)*)

Given a polynomial with rational coefficients, we can obtain an integer polynomial that differs from it only by a nonzero constant by clearing the denominators.

lemma *ratpoly-to-intpoly*:
assumes $\forall i. \text{poly.coeff } p \ i \in \mathbb{Q}$
obtains $q \ c$ **where** $c \neq 0 \ p = \text{Polynomial.smult } (\text{inverse } (\text{of-nat } c)) \ (\text{of-int-poly } q)$
proof (*cases p = 0*)
case *True*
with *that[of 1 0]* **show** *?thesis* **by** *auto*
next
case *False*
from *assms* **obtain** p' **where** $p' : p = \text{map-poly of-rat } p'$
using *ratpolyE* **by** *auto*
define c **where** $c = \text{Lcm } ((\text{nat} \circ \text{snd} \circ \text{quotient-of} \circ \text{poly.coeff } p') \ \{..\text{Polynomial.degree } p'\})$
have $\neg \text{snd } (\text{quotient-of } x) \leq 0$ **for** x
using *quotient-of-denom-pos[of x, OF surjective-pairing]* **by** *auto*
hence $c \neq 0$ **by** (*auto simp: c-def*)
define q **where** $q = \text{Polynomial.smult } (\text{of-nat } c) \ p'$

have $\text{poly.coeff } q \ i \in \mathbb{Z}$ **for** i
proof (*cases i > Polynomial.degree p'*)

```

case False
define m n
  where m = fst (quotient-of (poly.coeff p' i))
        and n = nat (snd (quotient-of (poly.coeff p' i)))
have mn: n > 0 poly.coeff p' i = of-int m / of-nat n
  using quotient-of-denom-pos[of poly.coeff p' i, OF surjective-pairing]
        quotient-of-div[of poly.coeff p' i, OF surjective-pairing]
  by (auto simp: m-def n-def)
from False have n dvd c unfolding c-def
  by (intro dvd-Lcm) (auto simp: c-def n-def o-def not-less)
hence of-nat c * (of-int m / of-nat n) = (of-nat (c div n) * of-int m :: rat)
  by (auto simp: of-nat-div)
also have ... ∈ ℤ by auto
finally show ?thesis using mn by (auto simp: q-def)
qed (auto simp: q-def coeff-eq-0)
with intpolyE obtain q' where q': q = of-int-poly q' by auto
moreover have p = Polynomial.smult (inverse (of-nat c)) (map-poly of-rat
(of-int-poly q^))
  unfolding smult-conv-map-poly q'[symmetric] p' using ⟨c ≠ 0⟩
  by (intro poly-eqI) (auto simp: coeff-map-poly q-def of-rat-mult)
ultimately show ?thesis
  using q' p' ⟨c ≠ 0⟩ by (auto intro!: that[of c q'])
qed

```

```

lemma symmetric-mpoly-symmetric-sum:
  assumes  $\bigwedge \pi. \pi \text{ permutes } A \implies g \pi \text{ permutes } X$ 
  assumes  $\bigwedge x \pi. x \in X \implies \pi \text{ permutes } A \implies \text{mpoly-map-vars } \pi (f x) = f (g \pi x)$ 
  shows symmetric-mpoly A ( $\sum_{x \in X}. f x$ )
  unfolding symmetric-mpoly-def
proof safe
  fix  $\pi$  assume  $\pi: \pi \text{ permutes } A$ 
  have  $\text{mpoly-map-vars } \pi (\text{sum } f X) = (\sum_{x \in X}. \text{mpoly-map-vars } \pi (f x))$ 
    by simp
  also have ... = ( $\sum_{x \in X}. f (g \pi x)$ )
    by (intro sum.cong assms  $\pi$  refl)
  also have ... = ( $\sum_{x \in g \pi' X}. f x$ )
    using assms(1)[OF  $\pi$ ] by (subst sum.reindex) (auto simp: permutes-inj-on)
  also have  $g \pi' X = X$ 
    using assms(1)[OF  $\pi$ ] by (simp add: permutes-image)
  finally show  $\text{mpoly-map-vars } \pi (\text{sum } f X) = \text{sum } f X$  .
qed

```

```

lemma symmetric-mpoly-symmetric-prod:
  assumes  $g \text{ permutes } X$ 
  assumes  $\bigwedge x \pi. x \in X \implies \pi \text{ permutes } A \implies \text{mpoly-map-vars } \pi (f x) = f (g x)$ 
  shows symmetric-mpoly A ( $\prod_{x \in X}. f x$ )
  unfolding symmetric-mpoly-def

```

```

proof safe
  fix  $\pi$  assume  $\pi$ :  $\pi$  permutes  $A$ 
  have  $\text{mpoly-map-vars } \pi$  ( $\text{prod } f X$ ) =  $(\prod_{x \in X}. \text{mpoly-map-vars } \pi (f x))$ 
    by simp
  also have  $\dots = (\prod_{x \in X}. f (g x))$ 
    by (intro prod.cong assms  $\pi$  refl)
  also have  $\dots = (\prod_{x \in g'X}. f x)$ 
    using assms by (subst prod.reindex) (auto simp: permutes-inj-on)
  also have  $g' X = X$ 
    using assms by (simp add: permutes-image)
  finally show  $\text{mpoly-map-vars } \pi$  ( $\text{prod } f X$ ) =  $\text{prod } f X$  .
qed

```

We now prove the transcendence of $i\pi$, from which the transcendence of π will follow as a trivial corollary. The first proof of this was given by von Lindemann [4]. The central ingredient is the fundamental theorem of symmetric functions.

The proof can, by now, be considered folklore and one can easily find many similar variants of it, but we mostly follows the nice exposition given by Niven [3].

An independent previous formalisation in Coq that uses the same basic techniques was given by Bernard et al. [2]. They later also formalised the much stronger Lindemann–Weierstraß theorem [1].

lemma *transcendental-i-pi*: $\neg \text{algebraic } (i * \pi)$

proof

— Suppose $i\pi$ were algebraic.

assume *algebraic* $(i * \pi)$

— We obtain some nonzero integer polynomial that has $i\pi$ as a root. We can assume w.l.o.g. that the constant coefficient of this polynomial is nonzero.

then obtain p

where p : *poly* (*of-int-poly* p) $(i * \pi) = 0$ $p \neq 0$ *poly.coeff* p $0 \neq 0$

by (*elim algebraicE'-nonzero*) *auto*

define n **where** $n = \text{Polynomial.degree } p$

— We define the sequence of the roots of this polynomial:

obtain *root* **where** *Polynomial.smult* (*Polynomial.lead-coeff* (*of-int-poly* p))

$(\prod_{i < n}. [:-\text{root } i :: \text{complex}, 1:]) = \text{of-int-poly } p$

using *complex-poly-decompose'*[*of of-int-poly p*] **unfolding** n -*def* **by** *auto*

note *root = this* [*symmetric*]

— We note that $i\pi$ is, of course, among these roots.

from p **and** *root* **obtain** idx **where** idx : $idx < n$ *root* $idx = i * \pi$

by (*auto simp: poly-prod*)

— We now define a new polynomial P' , whose roots are all numbers that arise as a sum of any subset of roots of p . We also count all those subsets that sum up to 0 and call their number A .

define $root'$ **where** $root' = (\lambda X. (\sum_{j \in X}. \text{root } j))$

define P **where** $P = (\lambda i. \prod X \mid X \subseteq \{..<n\} \wedge \text{card } X = i. [-\text{root}' X, 1:])$
define P' **where** $P' = (\prod i \in \{0 < .. n\}. P i)$
define A **where** $A = \text{card } \{X \in \text{Pow } \{..<n\}. \text{root}' X = 0\}$
have $[\text{simp}]$: $P' \neq 0$ **by** $(\text{auto simp: } P'\text{-def } P\text{-def})$

— We give the name Roots' to those subsets that do not sum to zero and note that there is at least one, namely $\{i\pi\}$.

define Roots' **where** $\text{Roots}' = \{X. X \subseteq \{..<n\} \wedge \text{root}' X \neq 0\}$
have $[\text{intro}]$: $\text{finite } \text{Roots}'$ **by** $(\text{auto simp: } \text{Roots}'\text{-def})$
have $\{idx\} \in \text{Roots}'$ **using** idx **by** $(\text{auto simp: } \text{Roots}'\text{-def } \text{root}'\text{-def})$
hence $\text{Roots}' \neq \{\}$ **by** auto
hence $\text{card-Roots}'$: $\text{card } \text{Roots}' > 0$ **by** $(\text{auto simp: } \text{card-eq-0-iff})$

have $P'\text{-altdef}$: $P' = (\prod X \in \text{Pow } \{..<n\} - \{\{\}\}. [-\text{root}' X, 1:])$

proof —

have $P' = (\prod (i, X) \in (\text{SIGMA } x: \{0 < .. n\}. \{X. X \subseteq \{..<n\} \wedge \text{card } X = x\}). [-\text{root}' X, 1:])$

unfolding $P'\text{-def } P\text{-def}$ **by** $(\text{subst prod.Sigma})$ auto

also have $\dots = (\prod X \in \text{Pow } \{..<n\} - \{\{\}\}. [-\text{root}' X, 1:])$

using card-mono $[\text{of } \{..<n\}]$

by $(\text{intro prod.reindex-bij-witness}[\text{of } - \lambda X. (\text{card } X, X) \lambda(-, X). X])$

$(\text{auto simp: case-prod-unfold card-gt-0-iff intro: finite-subset}[\text{of } - \{..<n\}])$

finally show $?thesis$.

qed

— Clearly, A is nonzero, since the empty set sums to 0.

have $A > 0$

proof —

have $\{\} \in \{X \in \text{Pow } \{..<n\}. \text{root}' X = 0\}$

by $(\text{auto simp: } \text{root}'\text{-def})$

thus $?thesis$ **by** $(\text{auto simp: } A\text{-def } \text{card-gt-0-iff})$

qed

— Since $e^{i\pi} + 1 = 0$, we know the following:

have $0 = (\prod i < n. \text{exp } (\text{root } i) + 1)$

using idx **by** force

— We rearrange this product of sums into a sum of products and collect all summands that are 1 into a separate sum, which we call A :

also have $\dots = (\sum X \in \text{Pow } \{..<n\}. \prod i \in X. \text{exp } (\text{root } i))$

by (subst prod-add) auto

also have $\dots = (\sum X \in \text{Pow } \{..<n\}. \text{exp } (\text{root}' X))$

by $(\text{intro sum.cong refl, subst exp-sum } [\text{symmetric}])$

$(\text{auto simp: } \text{root}'\text{-def intro: finite-subset}[\text{of } - \{..<n\}])$

also have $\text{Pow } \{..<n\} = \{X \in \text{Pow } \{..<n\}. \text{root}' X \neq 0\} \cup \{X \in \text{Pow } \{..<n\}. \text{root}' X = 0\}$

by auto

also have $(\sum X \in \dots. \text{exp } (\text{root}' X)) = (\sum X \mid X \subseteq \{..<n\} \wedge \text{root}' X \neq 0. \text{exp } (\text{root}' X)) +$

$(\sum X \mid X \subseteq \{..<n\} \wedge \text{root}' X = 0. \text{exp } (\text{root}' X))$

by (*subst sum.union-disjoint*) *auto*
also have $(\sum X \mid X \subseteq \{..<n\} \wedge \text{root}' X = 0. \text{exp} (\text{root}' X)) = \text{of-nat } A$
by (*simp add: A-def*)
— Finally, we obtain the fact that the sum of $\text{exp}(u)$ with u ranging over all the non-zero roots of P' is a negative integer.
finally have eq: $(\sum X \mid X \subseteq \{..<n\} \wedge \text{root}' X \neq 0. \text{exp} (\text{root}' X)) = -\text{of-nat } A$
by (*simp add: add-eq-0-iff2*)

— Next, we show that P' is a rational polynomial since it can be written as a symmetric polynomial expression (with rational coefficients) in the roots of p .
define *ratpolys* **where** *ratpolys* = $\{p :: \text{complex poly}. \forall i. \text{poly.coeff } p \ i \in \mathbf{Q}\}$
have *ratpolysI:* $p \in \text{ratpolys}$ **if** $\bigwedge i. \text{poly.coeff } p \ i \in \mathbf{Q}$ **for** p
using that by (*auto simp: ratpolys-def*)

have $P' \in \text{ratpolys}$
proof —
define *Pmv* :: *nat* \Rightarrow *complex poly mpoly*
where $Pmv = (\lambda i. \prod X \mid X \subseteq \{..<n\} \wedge \text{card } X = i. \text{Const } ([:0, 1:] - (\sum i \in X. \text{monom } (\text{Poly-Mapping.single } i \ 1) \ 1)))$
define *P'mv* **where** $P'mv = (\prod i \in \{0 <.. n\}. Pmv \ i)$
have *insertion* $(\lambda i. [: \text{root } i:]) P'mv \in \text{ratpolys}$
proof (*rule symmetric-poly-of-roots-in-subring* [**where** $l = \lambda x. [:x:]$])
show *ring-closed ratpolys*
by *standard* (*auto simp: ratpolys-def coeff-mult*)
then interpret *ring-closed ratpolys* .
show $\forall m. \text{coeff } P'mv \ m \in \text{ratpolys}$
by (*auto simp: P'mv-def Pmv-def coeff-monom when-def mpoly-coeff-Const coeff-pCons' ratpolysI*
intro!: coeff-prod-closed minus-closed sum-closed uminus-closed)
show $\forall i. [: \text{poly.coeff } (\text{of-int-poly } p) \ i:] \in \text{ratpolys}$
by (*intro ratpolysI allI*) (*auto simp: coeff-pCons'*)
show $[: \text{inverse } (\text{of-int } (\text{Polynomial.lead-coeff } p)):] * [: \text{of-int } (\text{Polynomial.lead-coeff } p) :: \text{complex}:] = 1$
using $\langle p \neq 0 \rangle$ **by** (*auto intro!: poly-eqI simp: field-simps*)
next
have *symmetric-mpoly* $\{..<n\}$ (*Pmv k*) **for** k
unfolding *symmetric-mpoly-def*
proof *safe*
fix $\pi :: \text{nat} \Rightarrow \text{nat}$ **assume** π *permutes* $\{..<n\}$
hence *mpoly-map-vars* π (*Pmv k*) =
 $(\prod X \mid X \subseteq \{..<n\} \wedge \text{card } X = k. \text{Const } [:0, 1:] - (\sum x \in X. \text{MPoly-Type.monom } (\text{Poly-Mapping.single } (\pi \ x) \ (\text{Suc } 0)))$
1))
by (*simp add: Pmv-def permutes-bij*)
also have $\dots = (\prod X \mid X \subseteq \{..<n\} \wedge \text{card } X = k. \text{Const } [:0, 1:] - (\sum x \in \pi' X. \text{MPoly-Type.monom } (\text{Poly-Mapping.single } x \ (\text{Suc } 0)))$
0)) 1))
using π **by** (*subst sum.reindex*) (*auto simp: permutes-inj-on*)
also have $\dots = (\prod X \in (\lambda X. \pi' X) \{X. X \subseteq \{..<n\} \wedge \text{card } X = k\}. \text{Const$

```

[:0, 1:] –
      (∑ x∈X. MPoly-Type.monom (Poly-Mapping.single x (Suc
0)) 1))
  by (subst prod.reindex) (auto intro!: inj-on-image permutes-inj-on[OF π])
  also have (λX. π 'X) '{X. X ⊆ {..

```

— We clear the denominators and remove all powers of X from P' to obtain a new integer polynomial Q .

```

define Q' where Q' = (∏ X∈Roots'. [: - root' X, 1:])
have P' = (∏ X∈Pow {..

```

have *degree-Q'*: $\text{Polynomial.degree } P' = \text{Polynomial.degree } Q' + (A - 1)$
by (*subst Q'*)
(auto simp: Q'-def Roots'-def degree-mult-eq Polynomial.degree-monom-eq degree-prod-sum-eq)

have $\forall i. \text{poly.coeff } Q' i \in \mathbb{Q}$
proof
fix $i :: \text{nat}$
have $\text{poly.coeff } Q' i = \text{Polynomial.coeff } P' (i + (A - 1))$
by (*simp add: Q' Polynomial.coeff-monom-mult*)
also have $\dots \in \mathbb{Q}$ **using** $\langle P' \in \text{ratpolys} \rangle$ **by** (*auto simp: ratpolys-def*)
finally show $\text{poly.coeff } Q' i \in \mathbb{Q}$.
qed
from *ratpoly-to-intpoly*[*OF this*] **obtain** $c Q$
where [*simp*]: $c \neq 0$ **and** $Q: Q' = \text{Polynomial.smult } (\text{inverse } (\text{of-nat } c))$
(*of-int-poly Q*)
by *metis*
have [*simp*]: $Q \neq 0$ **using** $Q Q'$ **by** *auto*
have $Q': \text{of-int-poly } Q = \text{Polynomial.smult } (\text{of-nat } c) Q'$
using Q **by** *simp*
have *degree-Q*: $\text{Polynomial.degree } Q = \text{Polynomial.degree } Q'$
by (*subst Q*) *auto*
have $\text{Polynomial.lead-coeff } (\text{of-int-poly } Q :: \text{complex poly}) = c$
by (*subst Q'*) (*simp-all add: degree-Q Q'-def Polynomial.lead-coeff-prod*)
hence *lead-coeff-Q*: $\text{Polynomial.lead-coeff } Q = \text{int } c$
using *of-int-eq-iff*[*of Polynomial.lead-coeff Q of-nat c*] **by** (*auto simp del: of-int-eq-iff*)
have *Q-decompose*: $\text{of-int-poly } Q =$
 $\text{Polynomial.smult } (\text{of-nat } c) (\prod_{X \in \text{Roots}'} [:- \text{root}' X, 1:])$
by (*subst Q'*) (*auto simp: Q'-def lead-coeff-Q*)
have $\text{poly } (\text{of-int-poly } Q) (i * \pi) = 0$
using $\langle \{idx\} \in \text{Roots}' \rangle \langle \text{finite Roots}' \rangle idx$
by (*force simp: root'-def Q-decompose poly-prod*)
have *degree-Q*: $\text{Polynomial.degree } (\text{of-int-poly } Q :: \text{complex poly}) = \text{card } \text{Roots}'$
by (*subst Q'*) (*auto simp: Q'-def degree-prod-sum-eq*)
have $\text{poly } (\text{of-int-poly } Q) (0 :: \text{complex}) \neq 0$
by (*subst Q'*) (*auto simp: Q'-def Roots'-def poly-prod*)
hence [*simp*]: $\text{poly } Q 0 \neq 0$ **by** *simp*
have [*simp*]: $\text{poly } (\text{of-int-poly } Q) (\text{root}' Y) = 0$ **if** $Y \in \text{Roots}'$ **for** Y
using *that* $\langle \text{finite Roots}' \rangle$ **by** (*auto simp: Q' Q'-def poly-prod*)

— We find some closed ball that contains all the roots of Q .

define r **where** $r = \text{Polynomial.degree } Q$
have $r > 0$ **using** *degree-Q card-Roots'* **by** (*auto simp: r-def*)
define *Radius* **where** $\text{Radius} = \text{Max } ((\lambda Y. \text{norm } (\text{root}' Y)) \text{ ` } \text{Roots}')$
have *Radius*: $\text{norm } (\text{root}' Y) \leq \text{Radius}$ **if** $Y \in \text{Roots}'$ **for** Y
using $\langle \text{finite Roots}' \rangle$ **that** **by** (*auto simp: Radius-def*)
from *Radius*[*of* $\{idx\}$] **have** $\text{Radius} \geq \pi$
using idx **by** (*auto simp: Roots'-def norm-mult root'-def*)

hence *Radius-nonneg*: $\text{Radius} \geq 0$ and $\text{Radius} > 0$ using *pi-gt3* by *linarith+*

— Since this ball is compact, Q is bounded on it. We obtain such a bound.

```

have compact (poly (of-int-poly Q :: complex poly) ‹ cball 0 Radius)
  by (intro compact-continuous-image continuous-intros) auto
then obtain Q-ub
  where Q-ub: Q-ub > 0
           $\wedge u :: \text{complex. } u \in \text{cball } 0 \text{ Radius} \implies \text{norm } (\text{poly } (\text{of-int-poly } Q) u)
\leq Q\text{-ub}$ 
```

by (auto dest!: compact-imp-bounded simp: bounded-pos cball-def)

— Using this, define another upper bound that we will need later.

```

define fp-ub
  where fp-ub =  $(\lambda p. |c| \wedge (r * p - 1) / \text{fact } (p - 1) * (\text{Radius} \wedge (p - 1) * Q\text{-ub} \wedge p)$ 
have fp-ub-nonneg: fp-ub p ≥ 0 for p
  unfolding fp-ub-def using ‹Radius ≥ 0› Q-ub
  by (intro mult-nonneg-nonneg divide-nonneg-pos zero-le-power) auto
define C where C = card Roots' * Radius * exp Radius
```

— We will now show that any sufficiently large prime number leads to $C * fp\text{-ub}$ $p \geq 1$, from which we will then derive a contradiction.

```

define primes-at-top where primes-at-top = inf-class.inf sequentially (principal {p. prime p})
have eventually  $(\lambda p. \forall x \in \{\text{nat } | \text{poly } Q \ 0|, c, A\}. p > x)$  sequentially
  by (intro eventually-ball-finite ballI eventually-gt-at-top) auto
hence eventually  $(\lambda p. \forall x \in \{\text{nat } | \text{poly } Q \ 0|, c, A\}. p > x)$  primes-at-top
  unfolding primes-at-top-def eventually-inf-principal by eventually-elim auto
moreover have eventually  $(\lambda p. \text{prime } p)$  primes-at-top
  by (auto simp: primes-at-top-def eventually-inf-principal)
ultimately have eventually  $(\lambda p. C * fp\text{-ub } p \geq 1)$  primes-at-top
proof eventually-elim
  case (elim p)
  hence p: prime p p > nat |poly Q 0| p > c p > A by auto
  hence p > 1 by (auto dest: prime-gt-1-nat)
```

— We define the polynomial $f(X) = \frac{c^s}{(p-1)!} X^{p-1} Q(X)^p$, where c is the leading coefficient of Q . We also define $F(X)$ to be the sum of all its derivatives.

```

define s where s = r * p - 1
define fp :: complex poly
  where fp = Polynomial.smult (of-nat c ^ s / fact (p - 1))
          (Polynomial.monom 1 (p - 1) * of-int-poly Q ^ p)
define Fp where Fp =  $(\sum i \leq s+p. (pderiv \wedge i) fp)$ 
define f F where f = poly fp and F = poly Fp
have degree-fp: Polynomial.degree fp = s + p using degree-Q card-Roots' ‹p > 1›
  by (simp add: fp-def s-def degree-mult-eq degree-monom-eq
        degree-power-eq r-def algebra-simps)
```

— Using the same argument as in the case of the transcendence of e , we now consider the function

$$I(u) := e^u F(0) - F(u) = u \int_0^1 e^{(1-t)x} f(tx) dt$$

whose absolute value can be bounded with a standard “maximum times length” estimate using our upper bound on f . All of this can be reused from the proof for e , so there is not much to do here. In particular, we will look at $\sum I(x_i)$ with the x_i ranging over the roots of Q and bound this sum in two different ways.

interpret *lindemann-weierstrass-aux fp* .

have *I-altdef*: $I = (\lambda u. \exp u * F 0 - F u)$

by (*intro ext*) (*simp add: I-def degree-fp F-def Fp-def poly-sum*)

— We show that *fp-ub* is indeed an upper bound for f .

have *fp-ub*: $\text{norm } (\text{poly } fp \ u) \leq \text{fp-ub } p$ **if** $u \in \text{cball } 0 \ \text{Radius}$ **for** u

proof —

have $\text{norm } (\text{poly } fp \ u) = |c|^{\wedge (r * p - 1)} / \text{fact } (p - 1) * (\text{norm } u)^{\wedge (p - 1)}$ *

$$\text{norm } (\text{poly } (\text{of-int-poly } Q) \ u)^{\wedge p}$$

by (*simp add: fp-def f-def s-def norm-mult poly-monom norm-divide norm-power*)

also have $\dots \leq \text{fp-ub } p$

unfolding *fp-ub-def* **using** that $Q\text{-ub } \langle \text{Radius} \geq 0 \rangle$

by (*intro mult-left-mono[OF mult-mono] power-mono zero-le-power*) *auto*

finally show *?thesis* .

qed

— We now show that the following sum is an integer multiple of p . This argument again uses the fundamental theorem of symmetric functions, exploiting that the inner sums are symmetric over the roots of Q .

have $(\sum i=p..s+p. \sum Y \in \text{Roots}'. \text{poly } ((\text{pderiv } \widehat{\wedge} i) \ \text{fp}) (\text{root}' \ Y)) / p \in \mathbb{Z}$

proof (*subst sum-divide-distrib, intro Ints-sum[of {a..b} for a b]*)

fix i **assume** $i: i \in \{p..s+p\}$

then obtain roots' **where** $\text{roots}': \text{distinct roots}' \ \text{set roots}' = \text{Roots}'$

using *finite-distinct-list* $\langle \text{finite } \text{Roots}' \rangle$ **by** *metis*

define l **where** $l = \text{length } \text{roots}'$

define fp' **where** $\text{fp}' = (\text{pderiv } \widehat{\wedge} i) \ \text{fp}$

define d **where** $d = \text{Polynomial.degree } \text{fp}'$

— We define a multivariate polynomial for the inner sum $\sum f(x_i)/p$ in order to show that it is indeed a symmetric function over the x_i .

define R **where** $R = (\text{smult } (1 / \text{of-nat } p) (\sum k \leq d. \sum i < l. \text{smult } (\text{poly.coeff } \text{fp}' \ k)$

$(\text{monom } (\text{Poly-Mapping.single } i \ k) (1 / \text{of-int } (c^{\wedge} k)))) ::$
complex mpoly)

— The j -th coefficient of the i -th derivative of f are integer multiples of $c^j p$ since $i \geq p$.

have *integer*: $\text{poly.coeff } \text{fp}' \ j / (\text{of-nat } c^{\wedge} j * \text{of-nat } p) \in \mathbb{Z}$ **if** $j \leq d$ **for** j

proof —

```

define fp'' where fp'' = Polynomial.monom 1 (p - 1) * Q ^ p
define x
  where x = c ^ s * poly.coeff ((pderiv ~ i) (Polynomial.monom 1 (p -
1) * Q ^ p)) j
  have [:fact p:] dvd ([:fact i:] :: int poly) using i
  by (auto intro: fact-dvd)
  also have [:fact i:] dvd ((pderiv ~ i) (Polynomial.monom 1 (p - 1) * Q
^ p))
  by (rule fact-dvd-higher-pderiv)
finally have c ^ j * fact p dvd x unfolding x-def of-nat-mult using that i
  by (intro mult-dvd-mono)
  (auto intro!: le-imp-power-dvd simp: s-def d-def fp'-def degree-higher-pderiv
degree-fp)
hence of-int x / (of-int (c ^ j * fact p) :: complex) ∈ ℤ
  by (intro of-int-divide-in-Ints) auto
also have of-int x / (of-int (c ^ j * fact p) :: complex) =
  poly.coeff fp' j / (of-nat c ^ j * of-nat p) using ⟨p > 1⟩
  unfolding x-def fp'-def fp-def
  by (simp add: fact-reduce[of p] field-simps hom-distribs higher-pderiv-smult
flip: of-int-hom.coeff-map-poly-hom)
finally show ?thesis .
qed

```

— Evaluating R yields is an integer since it is symmetric.

```

have insertion (λi. c * root' (roots' ! i)) R ∈ ℤ
proof (intro symmetric-poly-of-roots-in-subring-monic allI)
  define Q' where Q' = of-int-poly Q ∘p [:0, 1 / of-nat c :: complex:]
  show symmetric-mpoly {..<l} R unfolding R-def
  by (intro symmetric-mpoly-smult symmetric-mpoly-sum[of {..d}] symmet-
ric-mpoly-symmetric-sum)
  (simp-all add: mpoly-map-vars-monom permutes-bij permutep-single
bij-imp-bij-inv permutes-inv-inv)
  show MPoly-Type.coeff R m ∈ ℤ for m unfolding R-def coeff-sum coeff-smult
sum-distrib-left
  using integer by (auto simp: R-def coeff-monom when-def intro!: Ints-sum)
show vars R ⊆ {..<l} unfolding R-def
  by (intro order.trans[OF vars-smult] order.trans[OF vars-sum] UN-least
order.trans[OF vars-monom-subset]) auto
show ring-closed ℤ by standard auto

have (∏ i<l. [:- (of-nat c * root' (roots' ! i)), 1:]) =
  (∏ Y←roots'. [:- (of-nat c * root' Y), 1:])
  by (subst prod.list-conv-set-nth) (auto simp: atLeast0LessThan l-def)
also have ... = (∏ Y∈Roots'. [:- (of-nat c * root' Y), 1:])
  using roots' by (subst prod.distinct-set-conv-list [symmetric]) auto
also have ... = (∏ Y∈Roots'. Polynomial.smult (of-nat c) ([:-root' Y,
1:])) ∘p [:0, 1 / c:]
  by (simp add: pcompose-prod pcompose-pCons)
also have (∏ Y∈Roots'. Polynomial.smult (of-nat c) ([:-root' Y, 1:])) =

```

$Polynomial.smult (of\text{-}nat\ c \wedge card\ Roots') (\prod_{Y \in Roots'} [:-root'\ Y, 1:])$
 by (subst prod-smult) auto
 also have ... = $Polynomial.smult (of\text{-}nat\ c \wedge (card\ Roots' - 1))$
 $(Polynomial.smult\ c\ (\prod_{Y \in Roots'} [:-root'\ Y, 1:]))$
 using <finite Roots'> and <Roots' ≠ {}>
 by (subst power-diff) (auto simp: Suc-le-eq card-gt-0-iff)
 also have $Polynomial.smult\ c\ (\prod_{Y \in Roots'} [:-root'\ Y, 1:]) = of\text{-}int\text{-}poly$
 Q
 using $Q\text{-}decompose$ by simp
 finally show $Polynomial.smult (of\text{-}nat\ c \wedge (card\ Roots' - 1))\ Q' =$
 $(\prod_{i < l. [:- (of\text{-}nat\ c * root'\ (roots'!\ i)), 1:]})$
 by (simp add: pcompose-smult Q'-def)
 fix $i :: nat$
 show $poly.coeff (Polynomial.smult (of\text{-}nat\ c \wedge (card\ Roots' - 1))\ Q')\ i \in \mathbb{Z}$
 proof (cases i Polynomial.degree Q rule: linorder-cases)
 case greater
 thus ?thesis by (auto simp: Q'-def coeff-pcompose-linear coeff-eq-0)
 next
 case equal
 thus ?thesis using <Roots' ≠ {}> degree- Q card-Roots' lead-coeff- Q
 by (auto simp: Q'-def coeff-pcompose-linear lead-coeff- Q power-divide
 power-diff)
 next
 case less
 have $poly.coeff (Polynomial.smult (of\text{-}nat\ c \wedge (card\ Roots' - 1))\ Q')\ i =$
 $of\text{-}int (poly.coeff\ Q\ i) * (of\text{-}int (c \wedge (card\ Roots' - 1)) / of\text{-}int (c \wedge$
 $i))$
 by (auto simp: Q'-def coeff-pcompose-linear power-divide)
 also have ... $\in \mathbb{Z}$ using less degree- Q
 by (intro Ints-mult of-int-divide-in-Ints) (auto intro!: le-imp-power-dvd)
 finally show ?thesis .
 qed
 qed auto
 — Moreover, by definition, evaluating R gives us $\sum f(x_i)/p$.
 also have $insertion (\lambda i. c * root'\ (roots'!\ i))\ R =$
 $(\sum_{Y \leftarrow roots'} poly\ fp'\ (root'\ Y)) / of\text{-}nat\ p$
 by (simp add: insertion-sum R-def poly-altdef d-def sum-list-sum-nth
 atLeast0LessThan
 l-def power-mult-distrib algebra-simps
 sum.swap[of - {..Polynomial.degree fp'}] del: insertion-monom)
 also have ... = $(\sum_{Y \in Roots'} poly ((pderiv \widehat{\widehat{i}})\ fp)\ (root'\ Y)) / of\text{-}nat\ p$
 using roots' by (subst sum-list-distinct-conv-sum-set) (auto simp: fp'-def
 poly-pcompose)
 finally show ... $\in \mathbb{Z}$.
 qed
 then obtain K where $K: (\sum_{i=p..s+p} \sum_{Y \in Roots'} poly ((pderiv \widehat{\widehat{i}})\ fp)\ (root'\ Y)) = of\text{-}int\ K * p$
 using < $p > 1$ > by (auto elim!: Ints-cases simp: field-simps)

— Next, we show that $F(0)$ is an integer and coprime to p .

obtain $F0 :: \text{int}$ **where** $F0: F\ 0 = \text{of-int } F0 \text{ coprime } (\text{int } p) F0$

proof –

have $(\sum i=p..s + p. \text{poly } ((\text{pderiv } \sim i) \text{ fp})\ 0) / \text{of-nat } p \in \mathbb{Z}$

unfolding $\text{sum-divide-distrib}$

proof (intro Ints-sum)

fix i **assume** $i: i \in \{p..s+p\}$

hence fact $p \text{ dvd } \text{poly } ((\text{pderiv } \sim i) ([:0, 1:] \wedge (p - 1) * Q \wedge p))\ 0$

by ($\text{intro fact-dvd-poly-higher-pderiv-aux'}$) auto

then obtain k **where** $k: \text{poly } ((\text{pderiv } \sim i) ([:0, 1:] \wedge (p - 1) * Q \wedge p))$

$0 = k * \text{fact } p$

by auto

have $(\text{pderiv } \sim i) \text{ fp} = \text{Polynomial.smult } (\text{of-nat } c \wedge s / \text{fact } (p - 1))$

$(\text{of-int-poly } ((\text{pderiv } \sim i) ([:0, 1:] \wedge (p - 1) * Q \wedge p)))$

by ($\text{simp add: fp-def higher-pderiv-smult Polynomial.monom-altdef hom-distrib}$)

also have $\text{poly } \dots\ 0 / \text{of-nat } p = \text{of-int } (c \wedge s * k)$

using $k \langle p > 1 \rangle$ **by** ($\text{simp add: fact-reduce[of } p]$)

also have $\dots \in \mathbb{Z}$ **by** simp

finally show $\text{poly } ((\text{pderiv } \sim i) \text{ fp})\ 0 / \text{of-nat } p \in \mathbb{Z}$.

qed

then obtain S **where** $S: (\sum i=p..s + p. \text{poly } ((\text{pderiv } \sim i) \text{ fp})\ 0) = \text{of-int}$

$S * p$

using $\langle p > 1 \rangle$ **by** ($\text{auto elim!: Ints-cases simp: field-simps}$)

have $F\ 0 = (\sum i \leq s + p. \text{poly } ((\text{pderiv } \sim i) \text{ fp})\ 0)$

by ($\text{auto simp: F-def Fp-def poly-sum}$)

also have $\dots = (\sum i \in \text{insert } (p - 1) \{p..s + p\}. \text{poly } ((\text{pderiv } \sim i) \text{ fp})\ 0)$

proof ($\text{intro sum.mono-neutral-right ballI}$)

fix i **assume** $i: i \in \{..s + p\} - \text{insert } (p - 1) \{p..s + p\}$

hence $i < p - 1$ **by** auto

have $\text{Polynomial.monom } 1\ (p - 1) \text{ dvd } \text{fp}$

by ($\text{auto simp: fp-def intro: dvd-smult}$)

with i **show** $\text{poly } ((\text{pderiv } \sim i) \text{ fp})\ 0 = 0$

by ($\text{intro poly-higher-pderiv-aux1'[of - } p - 1]$) ($\text{auto simp: Polynomial.monom-altdef}$)

qed auto

also have $\dots = \text{poly } ((\text{pderiv } \sim (p - 1)) \text{ fp})\ 0 + \text{of-int } S * \text{of-nat } p$

using $\langle p > 1 \rangle S$ **by** (subst sum.insert) auto

also have $\text{poly } ((\text{pderiv } \sim (p - 1)) \text{ fp})\ 0 = \text{of-int } (c \wedge s * \text{poly } Q\ 0 \wedge p)$

using $\text{poly-higher-pderiv-aux2[of } p - 1\ 0\ \text{of-int-poly } Q \wedge p :: \text{complex poly}]$

by ($\text{simp add: fp-def higher-pderiv-smult Polynomial.monom-altdef}$)

finally have $F\ 0 = \text{of-int } (S * \text{int } p + c \wedge s * \text{poly } Q\ 0 \wedge p)$

by simp

moreover have $\text{coprime } p\ c \text{ coprime } (\text{int } p) (\text{poly } Q\ 0)$

using p **by** ($\text{auto intro!: prime-imp-coprime dest: dvd-imp-le-int[rotated]}$)

hence $\text{coprime } (\text{int } p) (c \wedge s * \text{poly } Q\ 0 \wedge p)$

by auto

hence *coprime* (*int* p) ($S * \text{int } p + c \wedge s * \text{poly } Q \ 0 \wedge p$)
unfolding *coprime-iff-gcd-eq-1 gcd-add-mult* **by** *auto*
ultimately show *?thesis using that*[*of* $S * \text{int } p + c \wedge s * \text{poly } Q \ 0 \wedge p$] **by**
blast
qed

— Putting everything together, we have shown that $\sum I(x_i)$ is an integer coprime to p , and therefore a nonzero integer, and therefore has an absolute value of at least 1.

have ($\sum_{Y \in \text{Roots}'}. I(\text{root}' Y)$) = $F \ 0 * (\sum_{Y \in \text{Roots}'}. \text{exp}(\text{root}' Y)) -$
 $(\sum_{Y \in \text{Roots}'}. F(\text{root}' Y))$

by (*simp add: I-altdef sum-subtractf sum-distrib-left sum-distrib-right algebra-simps*)

also have $\dots = -(\text{of-int}(F \ 0 * \text{int } A) +$
 $(\sum_{i \leq s+p}. \sum_{Y \in \text{Roots}'}. \text{poly}((\text{pderiv} \wedge i) \text{fp})(\text{root}' Y)))$

using $F \ 0$ **by** (*simp add: Roots'-def eq F-def Fp-def poly-sum sum.swap*[*of* $\{..s+p\}$])

also have ($\sum_{i \leq s+p}. \sum_{Y \in \text{Roots}'}. \text{poly}((\text{pderiv} \wedge i) \text{fp})(\text{root}' Y)$) =
 $(\sum_{i=p..s+p}. \sum_{Y \in \text{Roots}'}. \text{poly}((\text{pderiv} \wedge i) \text{fp})(\text{root}' Y))$

proof (*intro sum.mono-neutral-right ballI sum.neutral*)

fix $i \ Y$ **assume** $i \in \{..s+p\} - \{p..s+p\}$ **and** $Y: Y \in \text{Roots}'$

have $[-\text{root}' Y, 1:] \wedge p \ \text{dvd} \ \text{of-int-poly } Q \wedge p$

by (*intro dvd-power-same*) (*auto simp: dvd-iff-poly-eq-0 Y*)

hence $[-\text{root}' Y, 1:] \wedge p \ \text{dvd} \ \text{fp}$

by (*auto simp: fp-def intro!: dvd-smult*)

thus $\text{poly}((\text{pderiv} \wedge i) \text{fp})(\text{root}' Y) = 0$

using i **by** (*intro poly-higher-pderiv-aux1*) *auto*

qed *auto*

also have $\dots = \text{of-int}(K * \text{int } p)$ **using** K **by** *simp*

finally have ($\sum_{Y \in \text{Roots}'}. I(\text{root}' Y)$) = $-\text{of-int}(K * \text{int } p + F \ 0 * \text{int } A)$

by *simp*

moreover have *coprime* $p \ A$

using $p \ \langle A > 0 \rangle$ **by** (*intro prime-imp-coprime*) (*auto dest!: dvd-imp-le*)

hence *coprime* (*int* p) ($F \ 0 * \text{int } A$)

using $F \ 0$ **by** *auto*

hence *coprime* (*int* p) ($K * \text{int } p + F \ 0 * \text{int } A$)

using $F \ 0$ **unfolding** *coprime-iff-gcd-eq-1 gcd-add-mult* **by** *auto*

hence $K * \text{int } p + F \ 0 * \text{int } A \neq 0$

using p **by** (*intro notI*) *auto*

hence $\text{norm}(-\text{of-int}(K * \text{int } p + F \ 0 * \text{int } A) :: \text{complex}) \geq 1$

unfolding *norm-minus-cancel norm-of-int* **by** *linarith*

ultimately have $1 \leq \text{norm}(\sum_{Y \in \text{Roots}'}. I(\text{root}' Y))$ **by** *metis*

— The M-L bound on the integral gives us an upper bound:

also have $\text{norm}(\sum_{Y \in \text{Roots}'}. I(\text{root}' Y)) \leq$

$(\sum_{Y \in \text{Roots}'}. \text{norm}(\text{root}' Y) * \text{exp}(\text{norm}(\text{root}' Y))) * \text{fp-ub } p$

proof (*intro sum-norm-le lindemann-weierstrass-integral-bound fp-ub fp-ub-nonneg*)

fix $Y \ u$ **assume** $*$: $Y \in \text{Roots}' \ u \in \text{closed-segment } 0 \ (\text{root}' Y)$

hence *closed-segment* $0 \ (\text{root}' Y) \subseteq \text{cball } 0 \ \text{Radius}$

using $\langle \text{Radius} \geq 0 \rangle$ $\text{Radius}[\text{of } Y]$ **by** (*intro closed-segment-subset*) *auto*
with * **show** $u \in \text{cball } 0 \text{ Radius}$ **by** *auto*
qed
also have $\dots \leq (\sum_{Y \in \text{Roots}'} \text{Radius} * \exp(\text{Radius}) * \text{fp-ub } p)$
using Radius **by** (*intro sum-mono mult-right-mono mult-mono fp-ub-nonneg*
 $\langle \text{Radius} \geq 0 \rangle$) *auto*
also have $\dots = C * \text{fp-ub } p$ **by** (*simp add: C-def*)
finally show $1 \leq C * \text{fp-ub } p$.
qed

— It now only remains to show that this inequality is inconsistent for large p . This is obvious, since the upper bound is an exponential divided by a factorial and therefore clearly tends to zero.

have $(\lambda p. C * \text{fp-ub } p) \in \Theta(\lambda p. (C / (\text{Radius} * |c|)) * (p / 2^{\wedge} p) * ((2 * |c|^{\wedge} r * \text{Radius} * Q\text{-ub})^{\wedge} p / \text{fact } p))$
(is - $\in \Theta(?f)$) using *degree-Q card-Roots' $\langle \text{Radius} > 0 \rangle$*
by (*intro bigthetaI-cong eventually-mono[OF eventually-gt-at-top[of 0]]*)
(auto simp: fact-reduce power-mult [symmetric] r-def fp-ub-def power-diff power-mult-distrib)
also have $?f \in o(\lambda p. 1 * 1 * 1)$
proof (*intro landau-o.big-small-mult landau-o.big-mult*)
have $(\lambda x. (\text{real-of-int } (2 * |c|^{\wedge} r) * \text{Radius} * Q\text{-ub})^{\wedge} x / \text{fact } x) \longrightarrow 0$
by (*intro power-over-fact-tendsto-0*)
thus $(\lambda x. (\text{real-of-int } (2 * |c|^{\wedge} r) * \text{Radius} * Q\text{-ub})^{\wedge} x / \text{fact } x) \in o(\lambda x. 1)$
by (*intro smalloI-tendsto*) *auto*

qed *real-asymp+*

finally have $(\lambda p. C * \text{fp-ub } p) \in o(\lambda p. 1)$ **by** *simp*
from *smalloD-tendsto[OF this]* **have** $(\lambda p. C * \text{fp-ub } p) \longrightarrow 0$ **by** *simp*
hence *eventually* $(\lambda p. C * \text{fp-ub } p < 1)$ *at-top*
by (*intro order-tendstoD*) *auto*
hence *eventually* $(\lambda p. C * \text{fp-ub } p < 1)$ *primes-at-top*
unfolding *primes-at-top-def eventually-inf-principal* **by** *eventually-elim auto*
moreover note $\langle \text{eventually } (\lambda p. C * \text{fp-ub } p \geq 1) \text{ primes-at-top} \rangle$
— We therefore have a contradiction for any sufficiently large prime.
ultimately have *eventually* $(\lambda p. \text{False})$ *primes-at-top*
by *eventually-elim auto*

— Since sufficiently large primes always exist, this concludes the theorem.

moreover have *frequently* $(\lambda p. \text{prime } p)$ *sequentially*
using *primes-infinite* **by** (*simp add: cofinite-eq-sequentially[symmetric] Inf-many-def*)
ultimately show *False*
by (*auto simp: frequently-def eventually-inf-principal primes-at-top-def*)
qed

lemma *pcompose-conjugates-integer:*

assumes $\bigwedge i. \text{poly.coeff } p \ i \in \mathbb{Z}$

shows $\text{poly.coeff } (p \text{compose } p \ [:0, i:] * p \text{compose } p \ [:0, -i:]) \ i \in \mathbb{Z}$

proof —

let $?c = \lambda i. \text{poly.coeff } p \ i :: \text{complex}$

have $\text{poly.coeff } (pcompose\ p\ [:0, i:] * pcompose\ p\ [:0, -i:])\ i =$
 $i \wedge i * (\sum_{k \leq i}. (-1) \wedge (i - k) * ?c\ k * ?c\ (i - k))$
unfolding *coeff-mult sum-distrib-left*
by (*intro sum.cong*) (*auto simp: coeff-mult coeff-pcompose-linear power-minus'*
power-diff field-simps intro!: Ints-sum)
also have $(\sum_{k \leq i}. (-1) \wedge (i - k) * ?c\ k * ?c\ (i - k)) =$
 $(\sum_{k \leq i}. (-1) \wedge k * ?c\ k * ?c\ (i - k))$ (**is** $?S1 = ?S2$)
by (*intro sum.reindex-bij-witness*[*of - \lambda k. i - k \lambda k. i - k*]) (*auto simp: mult-ac*)
hence $?S1 = (?S1 + ?S2) / 2$ **by** *simp*
also have $\dots = (\sum_{k \leq i}. ((-1) \wedge k + (-1) \wedge (i - k)) / 2 * ?c\ k * ?c\ (i - k))$
by (*simp add: ring-distrib sum.distrib sum-divide-distrib [symmetric]*)
also have $\dots = (\sum_{k \leq i}. (1 + (-1) \wedge i) / 2 * (-1) \wedge k * ?c\ k * ?c\ (i - k))$
by (*intro sum.cong*) (*auto simp: power-add power-diff field-simps*)
also have $i \wedge i * \dots \in \mathbb{Z}$
proof (*cases even i*)
case *True*
thus $?thesis$
by (*intro Ints-mult Ints-sum assms*) (*auto elim!: evenE simp: power-mult*)
next
case *False*
hence $1 + (-1) \wedge i = (0 :: \text{complex})$ **by** (*auto elim!: oddE simp: power-mult*)
thus $?thesis$ **by** *simp*
qed
finally show $?thesis$.
qed

lemma algebraic-times-i:

assumes *algebraic x*
shows *algebraic (i * x) algebraic (-i * x)*
proof -
from *assms obtain p where p: poly p x = 0* $\forall i. \text{poly.coeff } p\ i \in \mathbb{Z}\ p \neq 0$
by (*auto elim!: algebraicE*)
define p' **where** $p' = pcompose\ p\ [:0, i:] * pcompose\ p\ [:0, -i:]$
have $p': \text{poly } p'\ (i * x) = 0\ \text{poly } p'\ (-i * x) = 0\ p' \neq 0$
by (*auto simp: p'-def poly-pcompose algebra-simps p pcompose-eq-0-iff dest: pcompose-eq-0*)
moreover have $\forall i. \text{poly.coeff } p'\ i \in \mathbb{Z}$
using p **unfolding** p' -*def* **by** (*intro allI pcompose-conjugates-integer*) *auto*
ultimately show *algebraic (i * x) algebraic (-i * x)* **by** (*intro algebraicI*[*of p'*];
simp)
qed

lemma algebraic-times-i-iff: *algebraic (i * x) \longleftrightarrow algebraic x*
using *algebraic-times-i*[*of x*] *algebraic-times-i*[*of i * x*] **by** *auto*

theorem transcendental-pi: $\neg \text{algebraic } \pi$
using *transcendental-i-pi* **by** (*simp add: algebraic-times-i-iff*)

end

References

- [1] S. Bernard. Formalization of the Lindemann-Weierstrass Theorem. In *Interactive Theorem Proving*, Brasilia, Brazil, Sept. 2017.
- [2] S. Bernard, Y. Bertot, L. Rideau, and P.-Y. Strub. Formal proofs of transcendence for e and pi as an application of multivariate and symmetric polynomials. In *Proceedings of the 5th ACM SIGPLAN Conference on Certified Programs and Proofs*, CPP 2016, pages 76–87, New York, NY, USA, 2016. ACM.
- [3] I. Niven. The transcendence of π . *The American Mathematical Monthly*, 46(8):469–471, 1939.
- [4] F. von Lindemann. Ueber die Zahl π . *Mathematische Annalen*, 20(2):213–225, Jun 1882.