

The Transcendence of π

Manuel Eberl

June 16, 2019

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	Preliminary facts	2
2	The Transcendence of π	10

1 Preliminary facts

```
theory Pi-Transcendental-Polynomial-Library
  imports HOL-Computational-Algebra.Computational-Algebra
begin
```

```
lemma Ints-sum: ( $\bigwedge x. x \in A \implies f x \in \mathbb{Z}$ )  $\implies$  sum f A  $\in \mathbb{Z}$ 
  by (induction A rule: infinite-finite-induct) auto
```

```
lemma Ints-prod: ( $\bigwedge x. x \in A \implies f x \in \mathbb{Z}$ )  $\implies$  prod f A  $\in \mathbb{Z}$ 
  by (induction A rule: infinite-finite-induct) auto
```

```
lemma sum-in-Rats [intro]: ( $\bigwedge x. x \in A \implies f x \in \mathbb{Q}$ )  $\implies$  sum f A  $\in \mathbb{Q}$ 
  by (induction A rule: infinite-finite-induct) auto
```

```
lemma prod-in-Rats [intro]: ( $\bigwedge x. x \in A \implies f x \in \mathbb{Q}$ )  $\implies$  prod f A  $\in \mathbb{Q}$ 
  by (induction A rule: infinite-finite-induct) auto
```

```
lemma poly-cnj: cnj (poly p z) = poly (map-poly cnj p) (cnj z)
  by (simp add: poly-altdef degree-map-poly coeff-map-poly)
```

```
lemma poly-cnj-real:
  assumes  $\bigwedge n. \text{poly.coeff } p \ n \in \mathbb{R}$ 
  shows cnj (poly p z) = poly p (cnj z)
proof -
  from assms have map-poly cnj p = p
    by (intro poly-eqI) (auto simp: coeff-map-poly Reals-cnj-iff)
  with poly-cnj[of p z] show ?thesis by simp
qed
```

```
lemma real-poly-cnj-root-iff:
  assumes  $\bigwedge n. \text{poly.coeff } p \ n \in \mathbb{R}$ 
  shows poly p (cnj z) = 0  $\iff$  poly p z = 0
proof -
  have poly p (cnj z) = cnj (poly p z)
    by (simp add: poly-cnj-real assms)
  also have ... = 0  $\iff$  poly p z = 0 by simp
  finally show ?thesis .
qed
```

```
lemma coeff-pcompose-linear:
  fixes p :: 'a :: comm-semiring-1 poly
  shows coeff (pcompose p [:0, c:]) i = c ^ i * coeff p i
  by (induction p arbitrary: i) (auto simp: pcompose-pCons coeff-pCons mult-ac
split: nat.splits)
```

```
lemma coeff-pCons': poly.coeff (pCons c p) n = (if n = 0 then c else poly.coeff p
(n - 1))
```

by *transfer'*(*auto split: nat.splits*)

lemma *prod-smult*: $(\prod x \in A. \text{Polynomial.smult } (c \ x) \ (p \ x)) = \text{Polynomial.smult } (prod \ c \ A) \ (prod \ p \ A)$
by (*induction A rule: infinite-finite-induct*) (*auto simp: mult-ac*)

lemma *degree-higher-pderiv*: $\text{Polynomial.degree } ((pderiv \ ^n) \ p) = \text{Polynomial.degree } p - n$
for $p :: 'a::\{\text{comm-semiring-1, semiring-no-zero-divisors, semiring-char-0}\}$ *poly*
by (*induction n*) (*auto simp: degree-pderiv*)

lemma *sum-to-poly*: $(\sum x \in A. [f \ x]) = [\sum x \in A. f \ x]$
by (*induction A rule: infinite-finite-induct*) *auto*

lemma *diff-to-poly*: $[c] - [d] = [c - d]$
by (*simp add: poly-eq-iff mult-ac*)

lemma *mult-to-poly*: $[c] * [d] = [c * d]$
by (*simp add: poly-eq-iff mult-ac*)

lemma *prod-to-poly*: $(\prod x \in A. [f \ x]) = [\prod x \in A. f \ x]$
by (*induction A rule: infinite-finite-induct*) (*auto simp: mult-to-poly mult-ac*)

lemma *coeff-mult-0*: $\text{poly.coeff } (p * q) \ 0 = \text{poly.coeff } p \ 0 * \text{poly.coeff } q \ 0$
by (*simp add: coeff-mult*)

lemma *card-poly-roots-bound*:
fixes $p :: 'a::\{\text{comm-ring-1, ring-no-zero-divisors}\}$ *poly*
assumes $p \neq 0$
shows $\text{card } \{x. \text{poly } p \ x = 0\} \leq \text{degree } p$
using *assms*
proof (*induction degree p arbitrary: p rule: less-induct*)
case (*less p*)
show *?case*
proof (*cases* $\exists x. \text{poly } p \ x = 0$)
case *False*
hence $\{x. \text{poly } p \ x = 0\} = \{\}$ **by** *blast*
thus *?thesis* **by** *simp*
next
case *True*
then obtain x **where** $\text{poly } p \ x = 0$ **by** *blast*
hence $[-x, 1:] \ \text{dvd } p$ **by** (*subst (asm) poly-eq-0-iff-dvd*)
then obtain q **where** $q: p = [-x, 1:] * q$ **by** (*auto simp: dvd-def*)
with $\langle p \neq 0 \rangle$ **have** [*simp*]: $q \neq 0$ **by** *auto*
have $\text{deg: degree } p = \text{Suc } (\text{degree } q)$
by (*subst q, subst degree-mult-eq*) *auto*
have $\text{card } \{x. \text{poly } p \ x = 0\} \leq \text{card } (\text{insert } x \ \{x. \text{poly } q \ x = 0\})$
by (*intro card-mono*) (*auto intro: poly-roots-finite simp: q*)
also have $\dots \leq \text{Suc } (\text{card } \{x. \text{poly } q \ x = 0\})$

by (rule card-insert-le-m1) auto
 also from deg have card {x. poly q x = 0} ≤ degree q
 using ⟨p ≠ 0⟩ and q by (intro less) auto
 also have Suc ... = degree p by (simp add: deg)
 finally show ?thesis by - simp-all
 qed
 qed

lemma poly-eqI-degree:

fixes p q :: 'a :: {comm-ring-1, ring-no-zero-divisors} poly
 assumes $\bigwedge x. x \in A \implies \text{poly } p \ x = \text{poly } q \ x$
 assumes card A > degree p card A > degree q
 shows p = q
 proof (rule ccontr)
 assume neq: p ≠ q
 have degree (p - q) ≤ max (degree p) (degree q)
 by (rule degree-diff-le-max)
 also from assms have ... < card A by linarith
 also have ... ≤ card {x. poly (p - q) x = 0}
 using neq and assms by (intro card-mono poly-roots-finite) auto
 finally have degree (p - q) < card {x. poly (p - q) x = 0} .
 moreover have degree (p - q) ≥ card {x. poly (p - q) x = 0}
 using neq by (intro card-poly-roots-bound) auto
 ultimately show False by linarith
 qed

lemma poly-root-order-induct [case-names 0 no-roots root]:

fixes p :: 'a :: idom poly
 assumes P 0 $\bigwedge p. (\bigwedge x. \text{poly } p \ x \neq 0) \implies P \ p$
 $\bigwedge p \ x \ n. n > 0 \implies \text{poly } p \ x \neq 0 \implies P \ p \implies P \ ([:-x, 1:] \wedge^n p)$
 shows P p
 proof (induction degree p arbitrary: p rule: less-induct)
 case (less p)
 consider p = 0 | p ≠ 0 $\exists x. \text{poly } p \ x = 0$ | $\bigwedge x. \text{poly } p \ x \neq 0$ by blast
 thus ?case
 proof cases
 case 3
 with assms(2)[of p] show ?thesis by simp
 next
 case 2
 then obtain x where x: poly p x = 0 by auto
 have [:-x, 1:] ^ order x p dvd p by (intro order-1)
 then obtain q where q: p = [:-x, 1:] ^ order x p * q by (auto simp: dvd-def)
 with 2 have [simp]: q ≠ 0 by auto
 have order-pos: order x p > 0
 using ⟨p ≠ 0⟩ and x by (auto simp: order-root)
 have order x p = order x p + order x q
 by (subst q, subst order-mult) (auto simp: order-power-n-n)
 hence [simp]: order x q = 0 by simp

have $deg: degree\ p = order\ x\ p + degree\ q$
by (*subst q, subst degree-mult-eq*) (*auto simp: degree-power-eq*)
with order-pos **have** $degree\ q < degree\ p$ **by** *simp*
hence $P\ q$ **by** (*rule less*)
with order-pos **have** $P\ ([: -x, 1:] ^ order\ x\ p * q)$
by (*intro assms(3)*) (*auto simp: order-root*)
with q **show** *?thesis* **by** *simp*
qed (*simp-all add: assms(1)*)
qed

lemma *complex-poly-decompose*:
 $smult\ (lead-coeff\ p)\ (\prod z | poly\ p\ z = 0.\ [: -z, 1:] ^ order\ z\ p) = (p :: complex\ poly)$
proof (*induction p rule: poly-root-order-induct*)
case (*no-roots p*)
show *?case*
proof (*cases degree p = 0*)
case *False*
hence $\neg constant\ (poly\ p)$ **by** (*subst constant-degree*)
with fundamental-theorem-of-algebra **and** *no-roots* **show** *?thesis* **by** *blast*
qed (*auto elim!: degree-eq-zeroE*)
next
case (*root p x n*)
from *root* **have** $*$: $\{z.\ poly\ ([: -x, 1:] ^ n * p)\ z = 0\} = insert\ x\ \{z.\ poly\ p\ z = 0\}$
by *auto*
have $smult\ (lead-coeff\ ([: -x, 1:] ^ n * p))\ (\prod z | poly\ ([: -x, 1:] ^ n * p)\ z = 0.\ [: -z, 1:] ^ order\ z\ ([: -x, 1:] ^ n * p)) =$
 $[: -x, 1:] ^ order\ x\ ([: -x, 1:] ^ n * p) * smult\ (lead-coeff\ p)\ (\prod z \in \{z.\ poly\ p\ z = 0\}.\ [: -z, 1:] ^ order\ z\ ([: -x, 1:] ^ n * p))$
by (*subst *, subst prod.insert*)
(*insert root, auto intro: poly-roots-finite simp: mult-ac lead-coeff-mult lead-coeff-power*)
also **have** $order\ x\ ([: -x, 1:] ^ n * p) = n$
using *root* **by** (*subst order-mult*) (*auto simp: order-power-n-n order-0I*)
also **have** $(\prod z \in \{z.\ poly\ p\ z = 0\}.\ [: -z, 1:] ^ order\ z\ ([: -x, 1:] ^ n * p)) =$
 $(\prod z \in \{z.\ poly\ p\ z = 0\}.\ [: -z, 1:] ^ order\ z\ p)$
proof (*intro prod.cong refl, goal-cases*)
case (*1 y*)
with *root* **have** $order\ y\ ([: -x, 1:] ^ n) = 0$ **by** (*intro order-0I*) *auto*
thus *?case* **using** *root* **by** (*subst order-mult*) *auto*
qed
also **note** *root.IH*
finally **show** *?case* .
qed *simp-all*

lemma *order-pos-iff*: $p \neq 0 \implies order\ a\ p > 0 \iff poly\ p\ a = 0$
using *order-root[of p a]* **by** *auto*

lift-definition *poly-roots-mset* :: ('a :: idom) poly \Rightarrow 'a multiset is
 $\lambda p x. \text{if } p = 0 \text{ then } 0 \text{ else Polynomial.order } x p$

proof –
fix *p* :: 'a poly
show $(\lambda x. \text{if } p = 0 \text{ then } 0 \text{ else order } x p) \in \text{multiset}$
by (cases *p* = 0)
(auto simp: multiset-def order-pos-iff intro: finite-subset[OF - poly-roots-finite[of *p*]])

qed

lemma *poly-roots-mset-0* [simp]: *poly-roots-mset* 0 = {#}
by transfer' auto

lemma *count-poly-roots-mset* [simp]:
 $p \neq 0 \implies \text{count } (\text{poly-roots-mset } p) a = \text{order } a p$
by transfer' auto

lemma *set-count-poly-roots-mset* [simp]:
 $p \neq 0 \implies \text{set-mset } (\text{poly-roots-mset } p) = \{x. \text{poly } p x = 0\}$
by (auto simp: set-mset-def order-pos-iff)

lemma *image-prod-mset-multiplicity*:
 $\text{prod-mset } (\text{image-mset } f M) = \text{prod } (\lambda x. f x \wedge \text{count } M x) (\text{set-mset } M)$

proof (induction *M*)
case (add *x M*)
show ?case
proof (cases $x \in \text{set-mset } M$)
case True
have $(\prod y \in \text{set-mset } (\text{add-mset } x M). f y \wedge \text{count } (\text{add-mset } x M) y) =$
 $(\prod y \in \text{set-mset } M. (\text{if } y = x \text{ then } f x \text{ else } 1) * f y \wedge \text{count } M y)$
using True add **by** (intro prod.cong) auto
also have $\dots = f x * (\prod y \in \text{set-mset } M. f y \wedge \text{count } M y)$
using True **by** (subst prod.distrib) auto
also note add.IH [symmetric]
finally show ?thesis **using** True **by** simp

next
case False
hence $(\prod y \in \text{set-mset } (\text{add-mset } x M). f y \wedge \text{count } (\text{add-mset } x M) y) =$
 $f x * (\prod y \in \text{set-mset } M. f y \wedge \text{count } (\text{add-mset } x M) y)$
by (auto simp: not-in-iff)
also have $(\prod y \in \text{set-mset } M. f y \wedge \text{count } (\text{add-mset } x M) y) =$
 $(\prod y \in \text{set-mset } M. f y \wedge \text{count } M y)$
using False **by** (intro prod.cong) auto
also note add.IH [symmetric]
finally show ?thesis **by** simp

qed
qed auto

lemma *complex-poly-decompose-multiset*:
 $smult (lead-coeff p) (\prod x \in \#poly\text{-}roots\text{-}mset p. [-x, 1:]) = (p :: complex\ poly)$
proof (*cases p = 0*)
 case *False*
 hence $(\prod x \in \#poly\text{-}roots\text{-}mset p. [-x, 1:]) = (\prod x \mid poly\ p\ x = 0. [-x, 1:] \wedge$
order x p)
 by (*subst image-prod-mset-multiplicity simp-all*)
 also have $smult (lead-coeff p) \dots = p$
 by (*rule complex-poly-decompose*)
 finally show *?thesis .*
qed *auto*

lemma (*in monoid-add*) *prod-list-prod-nth*:
 $prod\text{-}list\ xs = (\prod i = 0..<length\ xs. xs\ !\ i)$
proof –
 have $xs = map (\lambda i. xs\ !\ i) [0..<length\ xs]$
 by (*simp add: map-nth*)
 also have $prod\text{-}list\ \dots = (\prod i = 0..<length\ xs. xs\ !\ i)$
 by (*subst prod.distinct-set-conv-list [symmetric]*) *auto*
 finally show *?thesis .*
qed

lemma *prod-zero-iff'*: $finite\ A \implies prod\ f\ A = 0 \iff (\exists x \in A. f\ x = 0)$
for $f :: 'a \Rightarrow 'b :: \{comm\text{-}semiring\text{-}1, semiring\text{-}no\text{-}zero\text{-}divisors\}$
by (*induction A rule: infinite-finite-induct*) *auto*

lemma *degree-prod-eq*: $(\bigwedge x. x \in A \implies f\ x \neq 0) \implies degree\ (prod\ f\ A) = (\sum x \in A. degree\ (f\ x))$
for $f :: 'a \Rightarrow 'b :: \{comm\text{-}semiring\text{-}1, semiring\text{-}no\text{-}zero\text{-}divisors\}$ *poly*
by (*induction A rule: infinite-finite-induct*) (*auto simp: degree-mult-eq prod-zero-iff'*)

lemma *lead-coeff-prod*: $(\bigwedge x. x \in A \implies f\ x \neq 0) \implies lead\text{-}coeff\ (prod\ f\ A) = (\prod x \in A. lead\text{-}coeff\ (f\ x))$
for $f :: 'a \Rightarrow 'b :: \{comm\text{-}semiring\text{-}1, semiring\text{-}no\text{-}zero\text{-}divisors\}$ *poly*
by (*induction A rule: infinite-finite-induct*) (*auto simp: lead-coeff-mult prod-zero-iff'*)

lemma *complex-poly-decompose'*:
 obtains *root where* $smult (lead-coeff p) (\prod i < degree\ p. [-root\ i, 1:]) = (p :: complex\ poly)$
proof –
 obtain *roots where* $roots : mset\ roots = poly\text{-}roots\text{-}mset\ p$
 using *ex-mset by blast*

have $p = smult (lead-coeff p) (\prod x \in \#poly\text{-}roots\text{-}mset\ p. [-x, 1:])$
by (*rule complex-poly-decompose-multiset [symmetric]*)
also have $(\prod x \in \#poly\text{-}roots\text{-}mset\ p. [-x, 1:]) = (\prod x \leftarrow roots. [-x, 1:])$
by (*subst prod-mset-prod-list [symmetric]*) (*simp add: roots*)

also have $\dots = (\prod_{i < \text{length roots}} [:-\text{roots} ! i, 1:])$
by (*subst prod-list-prod-nth*) (*auto simp: atLeast0LessThan*)
finally have $\text{eq}: p = \text{smult} (\text{lead-coeff } p) (\prod_{i < \text{length roots}} [:-\text{roots} ! i, 1:])$.
also have [*simp*]: $\text{degree } p = \text{length roots}$
using *roots* **by** (*subst eq*) (*auto simp: degree-prod-eq*)
finally show *?thesis* **by** (*intro that[of $\lambda i. \text{roots} ! i$]*) *auto*
qed

lemma *rsquarefree-root-order*:
assumes *rsquarefree p poly p z = 0 p \neq 0*
shows $\text{order } z p = 1$
proof –
from *assms* **have** $\text{order } z p \in \{0, 1\}$ **by** (*auto simp: rsquarefree-def*)
moreover from *assms* **have** $\text{order } z p > 0$ **by** (*auto simp: order-root*)
ultimately show $\text{order } z p = 1$ **by** *auto*
qed

lemma *complex-poly-decompose-rsquarefree*:
assumes *rsquarefree p*
shows $\text{smult} (\text{lead-coeff } p) (\prod z | \text{poly } p z = 0. [:-z, 1:]) = (p :: \text{complex poly})$
proof (*cases p = 0*)
case *False*
have $(\prod z | \text{poly } p z = 0. [:-z, 1:]) = (\prod z | \text{poly } p z = 0. [:-z, 1:] ^ \text{order } z p)$
using *assms False* **by** (*intro prod.cong*) (*auto simp: rsquarefree-root-order*)
also have $\text{smult} (\text{lead-coeff } p) \dots = p$
by (*rule complex-poly-decompose*)
finally show *?thesis* .
qed *auto*

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 $\text{poly.coeff} (p \text{compose } p [:0, i] * p \text{compose } p [:0, -i]) i =$
 $i ^ i * (\sum_{k \leq i}. (-1) ^ (i - k) * ?c k * ?c (i - k))$
unfolding *coeff-mult sum-distrib-left*
by (*intro sum.cong*) (*auto simp: coeff-mult coeff-pcompose-linear power-minus'*
 $\text{power-diff field-simps intro!: Ints-sum}$)
also have $(\sum_{k \leq i}. (-1) ^ (i - k) * ?c k * ?c (i - k)) =$
 $(\sum_{k \leq i}. (-1) ^ k * ?c k * ?c (i - k))$ (**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 $\dots = (\sum_{k \leq i}. ((-1) ^ k + (-1) ^ (i - k)) / 2 * ?c k * ?c (i - k))$
by (*simp add: ring-distrib sum.distrib sum-divide-distrib [symmetric]*)
also have $\dots = (\sum_{k \leq i}. (1 + (-1) ^ i) / 2 * (-1) ^ k * ?c k * ?c (i - k))$
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** $p: \text{poly } p \ x = 0 \ \forall i. \text{coeff } p \ i \in \mathbb{Z} \ p \neq 0$
 by (*auto elim!: algebraicE*)
 define *p'* **where** $p' = \text{pcompose } p \ [:0, i:] * \text{pcompose } p \ [:0, -i:]$
 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 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*: $\text{algebraic } (i * x) \longleftrightarrow \text{algebraic } x$
 using *algebraic-times-i[of x] algebraic-times-i[of i * x]* **by** *auto*

lemma *ratpolyE*:
 assumes $\forall i. \text{poly.coeff } p \ i \in \mathbb{Q}$
 obtains *q* **where** $p = \text{map-poly of-rat } q$
proof –
 have $\forall i \in \{.. \text{Polynomial.degree } p\}. \exists x. \text{poly.coeff } p \ i = \text{of-rat } x$
 using *assms* **by** (*auto simp: Rats-def*)
 from *bchoice[OF this]* **obtain** *f*
 where $f: \bigwedge i. i \leq \text{Polynomial.degree } p \implies \text{poly.coeff } p \ i = \text{of-rat } (f \ i)$ **by** *blast*
 define *q* **where** $q = \text{Poly } (\text{map } f \ [0..<\text{Suc } (\text{Polynomial.degree } p)])$
 have $p = \text{map-poly of-rat } q$
 by (*intro poly-eqI*)
 (*auto simp: coeff-map-poly q-def nth-default-def f coeff-eq-0 simp del: upt-Suc*)
 with that show *?thesis* **by** *blast*
qed

end

2 The Transcendence of π

theory *Pi-Transcendental*

imports

E-Transcendental.E-Transcendental
Symmetric-Polynomials.Symmetric-Polynomials
HOL-Real-Asymp.Real-Asymp
Pi-Transcendental-Polynomial-Library

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 *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 int-polyE 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 mpoly-map-vars  $\pi$  (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 mpoly-map-vars  $\pi$  (sum f X) = sum f X .
qed

```

```

lemma symmetric-mpoly-symmetric-prod:
  assumes g 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 mpoly-map-vars  $\pi$  (prod  $f$   $X$ ) = ( $\prod_{x \in X}$ . 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 mpoly-map-vars  $\pi$  (prod  $f$   $X$ ) = 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–Weierstra theorem [1].

lemma *transcendental-i-pi*: \neg *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 = *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))

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

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

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

have $P' = (\prod_{(i, X) \in (SIGMA\ x:\{0 <..n\}). \{X. X \subseteq \{..<n\} \wedge card\ X = x\}}. [-root'\ X, 1:])$
unfolding $P'-def\ P-def$ **by** $(subst\ prod.Sigma)\ auto$
also have $\dots = (\prod_{X \in Pow\ \{..<n\} - \{\}}. [-root'\ X, 1:])$
using $card-mono[of\ \{..<n\}]$
by $(intro\ prod.reindex-bij-witness[of\ -\ \lambda X. (card\ X, X)\ \lambda(-, X). X])$
 $(auto\ simp: case-prod-unfold\ card-gt-0-iff\ intro: finite-subset[of\ -\ \{..<n\}])$
finally show $?thesis$.

qed

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

have $A > 0$
proof —
have $\{\} \in \{X \in Pow\ \{..<n\}. root'\ X = 0\}$
by $(auto\ simp: root'-def)$
thus $?thesis$ **by** $(auto\ simp: A-def\ card-gt-0-iff)$

qed

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

have $0 = (\prod_{i < n}. exp\ (root\ i) + 1)$
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 $\dots = (\sum_{X \in Pow\ \{..<n\}}. \prod_{i \in X}. exp\ (root\ i))$
by $(subst\ prod-add)\ auto$
also have $\dots = (\sum_{X \in Pow\ \{..<n\}}. exp\ (root'\ X))$
by $(intro\ sum.cong\ refl,\ subst\ exp-sum\ [symmetric])$
 $(auto\ simp: root'-def\ intro: finite-subset[of\ -\ \{..<n\}])$
also have $Pow\ \{..<n\} = \{X \in Pow\ \{..<n\}. root'\ X \neq 0\} \cup \{X \in Pow\ \{..<n\}. root'\ X = 0\}$
by $auto$
also have $(\sum_{X \in \dots} exp\ (root'\ X)) = (\sum_{X \mid X \subseteq \{..<n\} \wedge root'\ X \neq 0}. exp\ (root'\ X)) +$

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

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* = $\{p :: \text{complex poly}. \forall i. \text{poly.coeff } p \ i \in \mathbb{Q}\}$
have *ratpolysI*: $p \in \text{ratpolys}$ **if** $\bigwedge i. \text{poly.coeff } p \ i \in \mathbb{Q}$ **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** $\pi: \pi$ *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 $\dots = (\prod X \mid X \subseteq \{..<n\} \wedge \text{card } X = k. \text{Const } [:0, 1:] - (\sum x \in \pi' X. \text{MPoly-Type.monom } (\text{Poly-Mapping.single } x \ (\text{Suc } 0)) \ 1))$

```

    using  $\pi$  by (subst sum.reindex) (auto simp: permutes-inj-on)
    also have ... = ( $\prod X \in (\lambda X. \pi 'X) \{X. X \subseteq \{..<n\} \wedge \text{card } X = k\}$ . Const
[:0, 1:] -
      ( $\sum x \in X. \text{MPoly-Type.monom } (\text{Poly-Mapping.single } x \text{ (Suc }
0)) 1$ ))
    by (subst prod.reindex) (auto intro!: inj-on-image permutes-inj-on[OF  $\pi$ ])
    also have ( $\lambda X. \pi 'X \{X. X \subseteq \{..<n\} \wedge \text{card } X = k\} = \{X. X \subseteq \pi '
\{..<n\} \wedge \text{card } X = k\}$ )
    using  $\pi$  by (subst image-image-fixed-card-subset) (auto simp: permutes-inj-on)
    also have  $\pi ' \{..<n\} = \{..<n\}$ 
    by (intro permutes-image  $\pi$ )
    finally show  $\text{mpoly-map-vars } \pi (Pmv k) = Pmv k$  by (simp add: Pmv-def)
qed
thus symmetric-mpoly  $\{..<n\} P' mv$ 
  unfolding P' mv-def by (intro symmetric-mpoly-prod) auto
next
show vars-P' mv: vars P' mv  $\subseteq \{..<n\}$ 
  unfolding P' mv-def Pmv-def
  by (intro order.trans[OF vars-prod] UN-least order.trans[OF vars-diff]
    Un-least order.trans[OF vars-sum] order.trans[OF vars-monom-subset])
auto
qed (insert root, auto intro!: ratpolysI simp: coeff-pCons')
also have insertion ( $\lambda i. [:root i:]$ ) (Pmv k) = P k for k
  by (simp add: Pmv-def insertion-prod insertion-diff insertion-sum root'-def
P-def
      sum-to-poly del: insertion-monom)

hence insertion ( $\lambda i. [:root i:]$ ) P' mv = P'
  by (simp add: P' mv-def insertion-prod P'-def)
finally show P'  $\in$  ratpolys .
qed

```

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

```

define Q' where Q' = ( $\prod X \in \text{Roots}' . [: - \text{root}' X, 1:]$ )
have P' = ( $\prod X \in \text{Pow } \{..<n\} - \{\{\}\} . [: - \text{root}' X, 1:]$ )
  by (simp add: P'-altdef)
also have Pow  $\{..<n\} - \{\{\}\} = \text{Roots}' \cup
\{X. X \in \text{Pow } \{..<n\} - \{\{\}\} \wedge \text{root}' X = 0\}$  by (auto simp: root'-def
Roots'-def)
also have ( $\prod X \in \dots [: - \text{root}' X, 1:] =
Q' * [:0, 1:] ^ \text{card } \{X. X \subseteq \{..<n\} \wedge X \neq \{\} \wedge \text{root}' X = 0\}$ )
  by (subst prod.union-disjoint) (auto simp: Q'-def Roots'-def)
also have  $\{X. X \subseteq \{..<n\} \wedge X \neq \{\} \wedge \text{root}' X = 0\} = \{X. X \subseteq \{..<n\} \wedge
\text{root}' X = 0\} - \{\{\}\}$ 
  by auto
also have card ... = A - 1 unfolding A-def
  by (subst card-Diff-singleton) (auto simp: root'-def)
finally have Q': P' = Polynomial.monom 1 (A - 1) * Q'

```

by (*simp add: Polynomial.monom-altdef*)
have *degree-Q'*: *Polynomial.degree P' = Polynomial.degree Q' + (A - 1)*
by (*subst Q'*)
(auto simp: Q'-def Roots'-def degree-mult-eq Polynomial.degree-monom-eq degree-prod-eq)

have $\forall i. \text{poly.coeff } Q' i \in \mathbb{Q}$
proof
fix *i :: nat*
have *poly.coeff Q' i = 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 *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' = Polynomial.smult (inverse (of-nat c))*
(of-int-poly Q)
by *metis*
have [*simp*]: *Q $\neq 0$ using Q Q' by auto*
have *Q': of-int-poly Q = Polynomial.smult (of-nat c) Q'*
using *Q by simp*
have *degree-Q: Polynomial.degree Q = Polynomial.degree Q'*
by (*subst Q*) *auto*
have *Polynomial.lead-coeff (of-int-poly Q :: complex poly) = c*
by (*subst Q'*) (*simp-all add: degree-Q Q'-def lead-coeff-prod*)
hence *lead-coeff-Q: Polynomial.lead-coeff Q = 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: of-int-poly Q =*
Polynomial.smult (of-nat c) ($\prod_{X \in \text{Roots}' } [:- \text{root}' X, 1:]$)
by (*subst Q'*) (*auto simp: Q'-def lead-coeff-Q*)
have *poly (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: Polynomial.degree (of-int-poly Q :: complex poly) = card Roots'*
by (*subst Q'*) (*auto simp: Q'-def degree-prod-eq*)
have *poly (of-int-poly Q) (0 :: complex) $\neq 0$*
by (*subst Q'*) (*auto simp: Q'-def Roots'-def poly-prod*)
hence [*simp*]: *poly Q 0 $\neq 0$ by simp*
have [*simp*]: *poly (of-int-poly Q) (root' Y) = 0 if Y \in 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 = Polynomial.degree Q*
have *r > 0 using degree-Q card-Roots' by (auto simp: r-def)*
define *Radius* **where** *Radius = Max (($\lambda Y. \text{norm (root' Y)}$) ' Roots')*
have *Radius: norm (root' Y) \leq Radius if Y \in Roots' for Y*
using $\langle \text{finite Roots}' \rangle$ **that by (auto simp: Radius-def)**
from *Radius[of {idx}] have Radius $\geq pi$*

using *idx* **by** (*auto simp: Roots'-def norm-mult root'-def*)
hence *Radius-nonneg: Radius ≥ 0 and 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*
 $\bigwedge u :: \text{complex. } u \in \text{cball } 0 \text{ Radius} \implies \text{norm } (\text{poly } (\text{of-int-poly } Q) u)$
 $\leq Q\text{-ub}$
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 = (λp. |c| ^ (r * p - 1) / fact (p - 1) * (Radius ^ (p - 1) * Q-ub ^ 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 (λp. ∀ x ∈ {nat | poly Q 0|, c, A}. p > x) sequentially*
by (*intro eventually-ball-finite ballI eventually-gt-at-top*) *auto*
hence *eventually (λp. ∀ x ∈ {nat | 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 (λp. prime p) primes-at-top*
by (*auto simp: primes-at-top-def eventually-inf-principal*)
ultimately **have** *eventually (λp. C * fp-ub p ≥ 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 = (∑ i ≤ s+p. (pderiv ^ 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

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

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

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

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

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** roots' : *distinct roots' set roots' = Roots'*

using *finite-distinct-list* (*finite Roots'*) **by** *metis*

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

define fp' **where** $fp' = (pderiv \ \hat{\ } i) \ fp$

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

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

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

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

have *integer*: $\text{poly.coeff } fp' \ j / (\text{of-nat } c \ \hat{\ } j * \text{of-nat } p) \in \mathbf{Z}$ **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⟩
    by (auto simp: fact-reduce[of p] fp'-def fp-def higher-pderiv-smult x-def
field-simps
simp flip: coeff-of-int-poly higher-pderiv-of-int-poly)
  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}] symmetric-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-prod-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:]))) =

```

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

```

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  $(\sum_{i=p..s+p} \text{poly } ((\text{pderiv } ^i) \text{ fp})\ 0) / \text{of-nat } p \in \mathbb{Z}$ 
    unfolding sum-divide-distrib
  proof (intro Ints-sum)
    fix  $i$  assume  $i: i \in \{p..s+p\}$ 
    hence  $\text{fact } p \text{ dvd } \text{poly } ((\text{pderiv } ^i) ([:0, 1:] ^ (p - 1) * Q ^ p))\ 0$ 
      by (intro fact-dvd-poly-higher-pderiv-aux') auto
    then obtain  $k$  where  $k: \text{poly } ((\text{pderiv } ^i) ([:0, 1:] ^ (p - 1) * Q ^ p))$ 
       $0 = k * \text{fact } p$ 
      by auto

  have  $(\text{pderiv } ^i) \text{ fp} = \text{Polynomial.smult } (\text{of-nat } c ^ s / \text{fact } (p - 1))$ 
     $(\text{of-int-poly } ((\text{pderiv } ^i) ([:0, 1:] ^ (p - 1) * Q ^ p)))$ 
    by (simp add: fp-def higher-pderiv-smult Polynomial.monom-altdef
      flip: higher-pderiv-of-int-poly)
  also have  $\text{poly } \dots\ 0 / \text{of-nat } p = \text{of-int } (c ^ s * k)$ 
    using  $k \langle p > 1 \rangle$  by (simp add: fact-reduce[of p])
  also have  $\dots \in \mathbb{Z}$  by simp
  finally show  $\text{poly } ((\text{pderiv } ^i) \text{ fp})\ 0 / \text{of-nat } p \in \mathbb{Z}$  .
qed
then obtain  $S$  where  $S: (\sum_{i=p..s+p} \text{poly } ((\text{pderiv } ^i) \text{ fp})\ 0) = \text{of-int}$ 
 $S * p$ 
  using ⟨p > 1⟩ by (auto elim!: Ints-cases simp: field-simps)

have  $F\ 0 = (\sum_{i \leq s+p} \text{poly } ((\text{pderiv } ^i) \text{ fp})\ 0)$ 
  by (auto simp: F-def Fp-def poly-sum)
also have  $\dots = (\sum_{i \in \text{insert } (p - 1) \{p..s+p\}} \text{poly } ((\text{pderiv } ^i) \text{ fp})\ 0)$ 
proof (intro sum.mono-neutral-right ballI)
  fix  $i$  assume  $i: i \in \{\dots + p\} - \text{insert } (p - 1) \{p..s+p\}$ 
  hence  $i < p - 1$  by auto
  have  $\text{Polynomial.monom } 1\ (p - 1) \text{ dvd } \text{fp}$ 
    by (auto simp: fp-def intro: dvd-smult)
  with  $i$  show  $\text{poly } ((\text{pderiv } ^i) \text{ fp})\ 0 = 0$ 
    by (intro poly-higher-pderiv-aux1'[of - p - 1]) (auto simp: Polynomial.monom-altdef)
qed auto
also have  $\dots = \text{poly } ((\text{pderiv } ^{p-1}) \text{ fp})\ 0 + \text{of-int } S * \text{of-nat } p$ 
  using ⟨p > 1⟩  $S$  by (subst sum.insert) auto
also have  $\text{poly } ((\text{pderiv } ^{p-1}) \text{ fp})\ 0 = \text{of-int } (c ^ s * \text{poly } Q\ 0 ^ p)$ 
  using poly-higher-pderiv-aux2[of p - 1 0 of-int-poly Q ^ p :: complex poly]
  by (simp add: fp-def higher-pderiv-smult Polynomial.monom-altdef)
finally have  $F\ 0 = \text{of-int } (S * \text{int } p + c ^ s * \text{poly } Q\ 0 ^ p)$ 
  by simp
moreover have coprime  $p\ c$  coprime  $(\text{int } p)\ (\text{poly } Q\ 0)$ 
  using  $p$  by (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 $\text{coprime } (\text{int } p) (S * \text{int } p + c \wedge s * \text{poly } Q \ 0 \wedge p)$
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 $\dots = -(\text{of-int } (F \ 0 * \text{int } A) +$
 $(\sum_{i \leq s+p}. \sum_{Y \in \text{Roots}'}. \text{poly}((\text{pderiv} \wedge \wedge i) \text{fp})(\text{root}' Y)))$
using $F \ 0$ **by** (*simp add: Roots'-def eq F-def Fp-def poly-sum sum.swap*[*of -*
 $\{..s+p\}$])
also have $(\sum_{i \leq s+p}. \sum_{Y \in \text{Roots}'}. \text{poly}((\text{pderiv} \wedge \wedge i) \text{fp})(\text{root}' Y)) =$
 $(\sum_{i=p..s+p}. \sum_{Y \in \text{Roots}'}. \text{poly}((\text{pderiv} \wedge \wedge i) \text{fp})(\text{root}' Y))$
proof (*intro sum.mono-neutral-right ballI sum.neutral*)
fix $i \ Y$ **assume** $i: i \in \{..s+p\} - \{p..s+p\}$ **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 $\text{poly}((\text{pderiv} \wedge \wedge i) \text{fp})(\text{root}' Y) = 0$
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 $\text{coprime } p \ A$
using $p \ \langle A > 0 \rangle$ **by** (*intro prime-imp-coprime*) (*auto dest!: dvd-imp-le*)
hence $\text{coprime } (\text{int } p) (F \ 0 * \text{int } A)$
using $F \ 0$ **by** *auto*
hence $\text{coprime } (\text{int } p) (K * \text{int } p + F \ 0 * \text{int } A)$
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 (*root'* Y) \subseteq *cball* 0 *Radius*
using $\langle \text{Radius} \geq 0 \rangle$ *Radius*[*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

theorem *transcendental-pi: \neg 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.