The Hermite–Lindemann–WeierstraßTranscendence Theorem

Manuel Eberl

March 17, 2025

Abstract

This article provides a formalisation of the Hermite–Lindemann–WeierstraßTheorem (also known as simply Hermite–Lindemann or Lindemann–Weierstraß). This theorem is one of the crowning achievements of 19th century number theory.

The theorem states that if $\alpha_1, \ldots, \alpha_n \in \mathbb{C}$ are algebraic numbers that are linearly independent over $\mathbb{Z}$, then $e^{\alpha_1}, \ldots, e^{\alpha_n}$ are algebraically independent over $\mathbb{Q}$.

Like the previous formalisation in Coq by Bernard [2], I proceeded by formalising Baker's alternative formulation of the theorem [1] and then deriving the original one from that. Baker's version states that for any algebraic numbers $\beta_1, \ldots, \beta_n \in \mathbb{C}$ and distinct algebraic numbers $\alpha_i, \ldots, \alpha_n \in \mathbb{C}$, we have:

$$\beta_1 e^{\alpha_1} + \ldots + \beta_n e^{\alpha_n} = 0 \qquad \text{iff} \qquad \forall i.\ \beta_i = 0$$

This has a number of immediate corollaries, e.g.:

- e and π are transcendental
- e^z, $\sin z$, $\tan z$, etc. are transcendental for algebraic $z \in \mathbb{C} \setminus \{0\}$
- $\ln z$ is transcendental for algebraic $z \in \mathbb{C} \setminus \{0, 1\}$

Contents

1 Divisibility of algebraic integers

theory *Algebraic-Integer-Divisibility*
  imports *Algebraic-Numbers.Algebraic-Numbers*
begin

In this section, we define a notion of divisibility of algebraic integers: y is divisible by x if $y\ /\ x$ is an algebraic integer (or if x and y are both zero).

Technically, the definition does not require x and y to be algebraic integers themselves, but we will always use it that way (in fact, in our case x will always be a rational integer).

definition *alg-dvd* :: *'a* :: *field* ⇒ *'a* ⇒ *bool* (**infix** ‹*alg'-dvd*› *50*) **where**
  x alg-dvd y ⟷ (*x* = *0* ⟶ *y* = *0*) ∧ *algebraic-int* (*y* / *x*)

lemma *alg-dvd-imp-algebraic-int*:
  fixes *x y* :: *'a* :: *field-char-0*
  shows *x alg-dvd y* ⟹ *algebraic-int x* ⟹ *algebraic-int y*
  ⟨*proof*⟩

lemma *alg-dvd-0-left-iff* [*simp*]: *0 alg-dvd x* ⟷ *x* = *0*
  ⟨*proof*⟩

lemma *alg-dvd-0-right* [*iff*]: *x alg-dvd 0*
  ⟨*proof*⟩

lemma *one-alg-dvd-iff* [*simp*]: *1 alg-dvd x* ⟷ *algebraic-int x*
  ⟨*proof*⟩

lemma *alg-dvd-of-int* [*intro*]:
  assumes *x dvd y*
  shows *of-int x alg-dvd of-int y*
⟨*proof*⟩

lemma *alg-dvd-of-nat* [*intro*]:
  assumes *x dvd y*
  shows *of-nat x alg-dvd of-nat y*
  ⟨*proof*⟩

lemma *alg-dvd-of-int-iff* [*simp*]:
  (*of-int x* :: *'a* :: *field-char-0*) *alg-dvd of-int y* ⟷ *x dvd y*
⟨*proof*⟩

lemma *alg-dvd-of-nat-iff* [*simp*]:
  (*of-nat x* :: *'a* :: *field-char-0*) *alg-dvd of-nat y* ⟷ *x dvd y*
⟨*proof*⟩

lemma *alg-dvd-add* [*intro*]:
  fixes *x y z* :: *'a* :: *field-char-0*
  shows *x alg-dvd y* ⟹ *x alg-dvd z* ⟹ *x alg-dvd* (*y* + *z*)

⟨*proof*⟩

lemma *alg-dvd-uminus-right* [*intro*]: *x alg-dvd y* ⟹ *x alg-dvd* −*y*
⟨*proof*⟩

lemma *alg-dvd-uminus-right-iff* [*simp*]: *x alg-dvd* −*y* ⟷ *x alg-dvd y*
⟨*proof*⟩

lemma *alg-dvd-diff* [*intro*]:
fixes *x y z* :: *'a* :: *field-char-0*
shows *x alg-dvd y* ⟹ *x alg-dvd z* ⟹ *x alg-dvd* (*y* − *z*)
⟨*proof*⟩

lemma *alg-dvd-triv-left* [*intro*]: *algebraic-int y* ⟹ *x alg-dvd x* ∗ *y*
⟨*proof*⟩

lemma *alg-dvd-triv-right* [*intro*]: *algebraic-int x* ⟹ *y alg-dvd x* ∗ *y*
⟨*proof*⟩

lemma *alg-dvd-triv-left-iff*: *x alg-dvd x* ∗ *y* ⟷ *x* = *0* ∨ *algebraic-int y*
⟨*proof*⟩

lemma *alg-dvd-triv-right-iff*: *y alg-dvd x* ∗ *y* ⟷ *y* = *0* ∨ *algebraic-int x*
⟨*proof*⟩

lemma *alg-dvd-triv-left-iff'* [*simp*]: *x* ≠ *0* ⟹ *x alg-dvd x* ∗ *y* ⟷ *algebraic-int y*
⟨*proof*⟩

lemma *alg-dvd-triv-right-iff'* [*simp*]: *y* ≠ *0* ⟹ *y alg-dvd x* ∗ *y* ⟷ *algebraic-int x*
⟨*proof*⟩

lemma *alg-dvd-trans* [*trans*]:
fixes *x y z* :: *'a* :: *field-char-0*
shows *x alg-dvd y* ⟹ *y alg-dvd z* ⟹ *x alg-dvd z*
⟨*proof*⟩

lemma *alg-dvd-mono* [*simp*]:
fixes *a b c d* :: *'a* :: *field-char-0*
shows *a alg-dvd c* ⟹ *b alg-dvd d* ⟹ (*a* ∗ *b*) *alg-dvd* (*c* ∗ *d*)
⟨*proof*⟩

lemma *alg-dvd-mult* [*simp*]:
fixes *a b c* :: *'a* :: *field-char-0*
shows *a alg-dvd c* ⟹ *algebraic-int b* ⟹ *a alg-dvd* (*b* ∗ *c*)
⟨*proof*⟩

lemma *alg-dvd-mult2* [*simp*]:
fixes *a b c* :: *'a* :: *field-char-0*

shows *a alg-dvd b* $\Longrightarrow$ *algebraic-int c* $\Longrightarrow$ *a alg-dvd* $(b * c)$
⟨*proof*⟩

A crucial theorem: if an integer x divides a rational number y, then y is in fact also an integer, and that integer is a multiple of x.

lemma *alg-dvd-int-rat*:
fixes $y :: 'a ::$ *field-char-0*
assumes *of-int x alg-dvd y* **and** $y \in \mathbb{Q}$
shows $\exists n.\ y =$ *of-int n* $\wedge$ *x dvd n*
⟨*proof*⟩

lemma *prod-alg-dvd-prod*:
fixes $f :: 'a \Rightarrow 'b ::$ *field-char-0*
assumes $\bigwedge x.\ x \in A \Longrightarrow f\ x$ *alg-dvd g x*
shows *prod f A alg-dvd prod g A*
⟨*proof*⟩

lemma *alg-dvd-sum*:
fixes $f :: 'a \Rightarrow 'b ::$ *field-char-0*
assumes $\bigwedge x.\ x \in A \Longrightarrow y$ *alg-dvd f x*
shows *y alg-dvd sum f A*
⟨*proof*⟩

lemma *not-alg-dvd-sum*:
fixes $f :: 'a \Rightarrow 'b ::$ *field-char-0*
assumes $\bigwedge x.\ x \in A - \{x'\} \Longrightarrow y$ *alg-dvd f x*
assumes $\neg y$ *alg-dvd f x'*
assumes $x' \in A$ *finite A*
shows $\neg y$ *alg-dvd sum f A*
⟨*proof*⟩

lemma *fact-dvd-pochhammer*:
assumes $m \leq n + 1$
shows *fact m dvd pochhammer* $(int\ n - int\ m + 1)\ m$
⟨*proof*⟩

lemma *coeff-higher-pderiv*:
coeff $((pderiv\ \widehat{\ }\widehat{\ }\ m)\ f)\ n =$ *pochhammer* $(of\text{-}nat\ (Suc\ n))\ m *$ *coeff f* $(n + m)$
⟨*proof*⟩

lemma *fact-alg-dvd-poly-higher-pderiv*:
fixes $p :: 'a ::$ *field-char-0 poly*
assumes $\bigwedge i.$ *algebraic-int* $(poly.coeff\ p\ i)$ *algebraic-int x* $m \leq k$
shows *fact m alg-dvd poly* $((pderiv\ \widehat{\ }\widehat{\ }\ k)\ p)\ x$
⟨*proof*⟩

end

2 Auxiliary facts about univariate polynomials

theory *More-Polynomial-HLW*
imports
HOL−Computational-Algebra.Computational-Algebra
Polynomial-Factorization.Gauss-Lemma
Power-Sum-Polynomials.Power-Sum-Polynomials-Library
Algebraic-Numbers.Algebraic-Numbers
begin

instance *poly* :: ({*idom-divide*,*normalization-semidom-multiplicative*,*factorial-ring-gcd*,
semiring-gcd-mult-normalize}) *factorial-semiring-multiplicative*
⟨*proof*⟩

lemma *lead-coeff-prod-mset*:
fixes *A* :: *'a*::{*comm-semiring-1*, *semiring-no-zero-divisors*} *poly multiset*
shows *Polynomial.lead-coeff* (*prod-mset A*) = *prod-mset* (*image-mset Polynomial.lead-coeff A*)
⟨*proof*⟩

lemma *content-normalize* [*simp*]:
fixes *p* :: *'a* :: {*factorial-semiring*, *idom-divide*, *semiring-gcd*, *normalization-semidom-multiplicative*} *poly*
shows *content* (*normalize p*) = *content p*
⟨*proof*⟩

lemma *rat-to-normalized-int-poly-exists*:
fixes *p* :: *rat poly*
assumes *p* ≠ *0*
obtains *q lc* **where** *p* = *Polynomial.smult lc* (*of-int-poly q*) *lc* > *0 content q* = *1*
⟨*proof*⟩

lemma *irreducible-imp-squarefree*:
assumes *irreducible p*
shows *squarefree p*
⟨*proof*⟩

lemma *squarefree-imp-rsquarefree*:
fixes *p* :: *'a* :: *idom poly*
assumes *squarefree p*
shows *rsquarefree p*
⟨*proof*⟩

lemma *squarefree-imp-coprime-pderiv*:
fixes *p* :: *'a* :: {*factorial-ring-gcd*,*semiring-gcd-mult-normalize*,*semiring-char-0*} *poly*
assumes *squarefree p* **and** *content p* = *1*
shows *Rings.coprime p* (*pderiv p*)

⟨*proof*⟩

lemma *irreducible-imp-coprime-pderiv*:
fixes *p* :: *'a* :: {*idom-divide*,*semiring-char-0*} *poly*
assumes *irreducible p Polynomial.degree p* ≠ *0*
shows *Rings.coprime p* (*pderiv p*)
⟨*proof*⟩

lemma *poly-gcd-eq-0I*:
assumes *poly p x* = *0 poly q x* = *0*
shows *poly* (*gcd p q*) *x* = *0*
⟨*proof*⟩

lemma *poly-eq-0-coprime*:
assumes *Rings.coprime p q p* ≠ *0 q* ≠ *0*
shows *poly p x* ≠ *0* ∨ *poly q x* ≠ *0*
⟨*proof*⟩

lemma *coprime-of-int-polyI*:
assumes *Rings.coprime p q*
shows *Rings.coprime* (*of-int-poly p*) (*of-int-poly q* :: *'a* :: {*field-char-0*,*field-gcd*} *poly*)
⟨*proof*⟩

lemma *irreducible-imp-rsquarefree-of-int-poly*:
fixes *p* :: *int poly*
assumes *irreducible p* **and** *Polynomial.degree p* > *0*
shows *rsquarefree* (*of-int-poly p* :: *'a* :: {*field-gcd*, *field-char-0*} *poly*)
⟨*proof*⟩

lemma *squarefree-of-int-polyI*:
assumes *squarefree p content p* = *1*
shows *squarefree* (*of-int-poly p* :: *'a* :: {*field-char-0*,*field-gcd*} *poly*)
⟨*proof*⟩

lemma *higher-pderiv-pcompose-linear*:
(*pderiv* ^^ *n*) (*pcompose p* [:*0*, *c*:]) =
Polynomial.smult (*c* ^ *n*) (*pcompose* ((*pderiv* ^^ *n*) *p*) [:*0*, *c*:])
⟨*proof*⟩

lemma *poly-poly-eq*:
poly (*poly p* [:*x*:]) *y* = *poly* (*eval-poly* (λ*p*. [:*poly p y*:]) *p* [:*0*, *1*:]) *x*
⟨*proof*⟩

lemma *poly-poly-poly-y-x* [*simp*]:
fixes *p* :: *'a* :: *idom poly poly*
shows *poly* (*poly* (*poly-y-x p*) [:*y*:]) *x* = *poly* (*poly p* [:*x*:]) *y*
⟨*proof*⟩

lemma (**in** *idom-hom*) *map-poly-higher-pderiv* [*hom-distribs*]:
map-poly hom ((*pderiv* ^^ *n*) *p*) = (*pderiv* ^^ *n*) (*map-poly hom p*)
⟨*proof*⟩

lemma *coeff-prod-linear-factors*:
fixes *f* :: *'a* ⇒ *'b* :: *comm-ring-1*
assumes [*intro*]: *finite A*
shows *Polynomial.coeff* (∏ *x*∈*A*. [:−*f x*, *1*:] ^ *e x*) *i* =
(∑ *X* | *X* ∈ *Pow* (*SIGMA x*:*A*. {..<*e x*}) ∧ *i* = *sum e A* − *card X*.
(−*1*) ^ *card X* ∗ (∏ *x*∈*X*. *f* (*fst x*)))
⟨*proof*⟩

lemma (**in** *comm-ring-hom*) *synthetic-div-hom*:
synthetic-div (*map-poly hom p*) (*hom x*) = *map-poly hom* (*synthetic-div p x*)
⟨*proof*⟩

lemma *synthetic-div-altdef*:
fixes *p* :: *'a* :: *field poly*
shows *synthetic-div p c* = *p div* [:−*c*, *1*:]
⟨*proof*⟩

lemma (**in** *ring-closed*) *poly-closed* [*intro*]:
assumes ⋀*i*. *poly.coeff p i* ∈ *A x* ∈ *A*
shows *poly p x* ∈ *A*
⟨*proof*⟩

lemma (**in** *ring-closed*) *coeff-pCons-closed* [*intro*]:
assumes ⋀*i*. *poly.coeff p i* ∈ *A x* ∈ *A*
shows *poly.coeff* (*pCons x p*) *i* ∈ *A*
⟨*proof*⟩

lemma (**in** *ring-closed*) *coeff-poly-mult-closed* [*intro*]:
assumes ⋀*i*. *poly.coeff p i* ∈ *A* ⋀*i*. *poly.coeff q i* ∈ *A*
shows *poly.coeff* (*p* ∗ *q*) *i* ∈ *A*
⟨*proof*⟩

lemma (**in** *ring-closed*) *coeff-poly-prod-closed* [*intro*]:
assumes ⋀*x i*. *x* ∈ *X* ⟹ *poly.coeff* (*f x*) *i* ∈ *A*
shows *poly.coeff* (*prod f X*) *i* ∈ *A*
⟨*proof*⟩

lemma (**in** *ring-closed*) *coeff-poly-power-closed* [*intro*]:
assumes ⋀*i*. *poly.coeff p i* ∈ *A*
shows *poly.coeff* (*p* ^ *n*) *i* ∈ *A*
⟨*proof*⟩

lemma (**in** *ring-closed*) *synthetic-div-closed*:
assumes ⋀*i*. *poly.coeff p i* ∈ *A x* ∈ *A*
shows *poly.coeff* (*synthetic-div p x*) *i* ∈ *A*

⟨*proof*⟩

lemma *pcompose-monom*: *pcompose* (*Polynomial.monom c n*) *p* = *Polynomial.smult c* (*p* ^ *n*)
⟨*proof*⟩

lemma *poly-roots-uminus* [*simp*]: *poly-roots* (−*p*) = *poly-roots p*
⟨*proof*⟩

lemma *poly-roots-normalize* [*simp*]:
fixes *p* :: *'a* :: {*normalization-semidom*, *idom-divide*} *poly*
shows *poly-roots* (*normalize p*) = *poly-roots p*
⟨*proof*⟩

lemma *poly-roots-of-int-normalize* [*simp*]:
poly-roots (*of-int-poly* (*normalize p*) :: *'a* :: {*idom*, *ring-char-0*} *poly*) =
poly-roots (*of-int-poly p*)
⟨*proof*⟩

lemma *poly-roots-power* [*simp*]: *poly-roots* (*p* ^ *n*) = *repeat-mset n* (*poly-roots p*)
⟨*proof*⟩

lemma *poly-roots-conv-sum-prime-factors*:
poly-roots q = (∑ *p*∈#*prime-factorization q*. *poly-roots p*)
⟨*proof*⟩

lemma *poly-roots-of-int-conv-sum-prime-factors*:
poly-roots (*of-int-poly q* :: *'a* :: {*idom*, *ring-char-0*} *poly*) =
(∑ *p*∈#*prime-factorization q*. *poly-roots* (*of-int-poly p*))
⟨*proof*⟩

lemma *dvd-imp-poly-roots-subset*:
assumes *q* ≠ *0 p dvd q*
shows *poly-roots p* ⊆# *poly-roots q*
⟨*proof*⟩

lemma *abs-prod-mset*: |*prod-mset* (*A* :: *'a* :: *idom-abs-sgn multiset*)| = *prod-mset* (*image-mset abs A*)
⟨*proof*⟩

lemma *content-1-imp-nonconstant-prime-factors*:
assumes *content* (*p* :: *int poly*) = *1* **and** *q* ∈ *prime-factors p*
shows *Polynomial.degree q* > *0*
⟨*proof*⟩

end

This theory imports both univariate and multivariate polynomials and thereby causes several overlaps in notation of polynomials.

theory *More-Min-Int-Poly*
imports
Algebraic-Numbers.Min-Int-Poly
HOL−Computational-Algebra.Computational-Algebra
More-Polynomial-HLW
begin

lemma *min-int-poly-squarefree* [*intro*]:
fixes $x :: 'a :: \{field\text{-}char\text{-}0, field\text{-}gcd\}$
shows *squarefree* (*min-int-poly x*)
⟨*proof*⟩

lemma *min-int-poly-conv-Gcd*:
fixes $x :: 'a :: \{field\text{-}char\text{-}0, field\text{-}gcd\}$
assumes *algebraic x*
shows *min-int-poly x* = *Gcd* $\{p.\ p \neq 0 \wedge p\ represents\ x\}$
⟨*proof*⟩

end

3 The lexicographic ordering on complex numbers

theory *Complex-Lexorder*
imports *Complex-Main HOL−Library.Multiset*
begin

We define a lexicographic order on the complex numbers, comparing first the real parts and, if they are equal, the imaginary parts. This ordering is of course not compatible with multiplication, but it is compatible with addition.

definition *less-eq-complex-lex* (**infix** $\langle\leq_{\mathbb{C}}\rangle$ *50*) **where**
less-eq-complex-lex x y ⟷ *Re x* < *Re y* ∨ *Re x* = *Re y* ∧ *Im x* ≤ *Im y*

definition *less-complex-lex* (**infix** $\langle<_{\mathbb{C}}\rangle$ *50*) **where**
less-complex-lex x y ⟷ *Re x* < *Re y* ∨ *Re x* = *Re y* ∧ *Im x* < *Im y*

interpretation *complex-lex*:
linordered-ab-group-add (+) *0* (−) *uminus less-eq-complex-lex less-complex-lex*
⟨*proof*⟩

lemmas [*trans*] =
complex-lex.order.trans complex-lex.less-le-trans
complex-lex.less-trans complex-lex.le-less-trans

lemma (**in** *ordered-comm-monoid-add*) *sum-mono-complex-lex*:
$(\bigwedge i.\ i{\in}K \Longrightarrow f\ i \leq_{\mathbb{C}} g\ i) \Longrightarrow (\sum i{\in}K.\ f\ i) \leq_{\mathbb{C}} (\sum i{\in}K.\ g\ i)$
⟨*proof*⟩

lemma *sum-strict-mono-ex1-complex-lex*:
fixes $f\ g :: 'i \Rightarrow complex$
assumes *finite A*
and $\forall x{\in}A.\ f\ x \leq_{\mathbb{C}} g\ x$
and $\exists a{\in}A.\ f\ a <_{\mathbb{C}} g\ a$
shows $sum\ f\ A <_{\mathbb{C}} sum\ g\ A$
⟨*proof*⟩

lemma *sum-list-mono-complex-lex*:
assumes $list\text{-}all2\ (\leq_{\mathbb{C}})\ xs\ ys$
shows $sum\text{-}list\ xs \leq_{\mathbb{C}} sum\text{-}list\ ys$
⟨*proof*⟩

lemma *sum-mset-mono-complex-lex*:
assumes $rel\text{-}mset\ (\leq_{\mathbb{C}})\ A\ B$
shows $sum\text{-}mset\ A \leq_{\mathbb{C}} sum\text{-}mset\ B$
⟨*proof*⟩

lemma *rel-msetI*:
assumes $list\text{-}all2\ R\ xs\ ys\ mset\ xs = A\ mset\ ys = B$
shows $rel\text{-}mset\ R\ A\ B$
⟨*proof*⟩

lemma *mset-replicate* [*simp*]: $mset\ (replicate\ n\ x) = replicate\text{-}mset\ n\ x$
⟨*proof*⟩

lemma *rel-mset-replicate-mset-right*:
assumes $\bigwedge x.\ x \in\# A \Longrightarrow R\ x\ y\ size\ A = n$
shows $rel\text{-}mset\ R\ A\ (replicate\text{-}mset\ n\ y)$
⟨*proof*⟩

end

4 Additional facts about multivariate polynomials

theory *More-Multivariate-Polynomial-HLW*
imports *Power-Sum-Polynomials.Power-Sum-Polynomials-Library*
begin

4.1 Miscellaneous

lemma *Var-altdef*: $Var\ i = monom\ (Poly\text{-}Mapping.single\ i\ 1)\ 1$
⟨*proof*⟩

lemma *Const-conv-monom*: $Const\ c = monom\ 0\ c$
⟨*proof*⟩

lemma *smult-conv-mult-Const*: $smult\ c\ p = Const\ c * p$
⟨*proof*⟩

lemma *mpoly-map-vars-Var* [*simp*]: *bij f* $\Longrightarrow$ *mpoly-map-vars f* (*Var i*) = *Var* (*f i*)
⟨*proof*⟩

lemma *symmetric-mpoly-symmetric-prod'*:
assumes $\bigwedge\pi$. π *permutes A* $\Longrightarrow$ *g* π *permutes X*
assumes $\bigwedge x\ \pi$. *x* $\in$ *X* $\Longrightarrow$ π *permutes A* $\Longrightarrow$ *mpoly-map-vars* π (*f x*) = *f* (*g* π *x*)
shows *symmetric-mpoly A* ($\prod x{\in}X.$ *f x*)
⟨*proof*⟩

4.2 Converting a univariate polynomial into a multivariate one

lift-definition *mpoly-of-poly-aux* :: *nat* $\Rightarrow$ *'a* :: *zero poly* $\Rightarrow$ (*nat* $\Rightarrow_0$ *nat*) $\Rightarrow_0$ *'a* **is**
$\lambda i\ c\ m$. *if Poly-Mapping.keys m* $\subseteq$ $\{i\}$ *then c* (*Poly-Mapping.lookup m i*) *else 0*
⟨*proof*⟩

lift-definition *mpoly-of-poly* :: *nat* $\Rightarrow$ *'a* :: *zero poly* $\Rightarrow$ *'a mpoly* **is**
mpoly-of-poly-aux ⟨*proof*⟩

lemma *mpoly-of-poly-0* [*simp*]: *mpoly-of-poly i 0* = *0*
⟨*proof*⟩

lemma *coeff-mpoly-of-poly1* [*simp*]:
coeff (*mpoly-of-poly i p*) (*Poly-Mapping.single i n*) = *poly.coeff p n*
⟨*proof*⟩

lemma *coeff-mpoly-of-poly2* [*simp*]:
assumes $\neg$*keys x* $\subseteq$ $\{i\}$
shows *coeff* (*mpoly-of-poly i p*) *x* = *0*
⟨*proof*⟩

lemma *coeff-mpoly-of-poly*:
coeff (*mpoly-of-poly i p*) *m* =
(*poly.coeff p* (*Poly-Mapping.lookup m i*) *when keys m* $\subseteq$ $\{i\}$)
⟨*proof*⟩

lemma *poly-mapping-single-eq-0-iff* [*simp*]: *Poly-Mapping.single i n* = *0* $\longleftrightarrow$ *n* = *0*
⟨*proof*⟩

lemma *mpoly-of-poly-pCons* [*simp*]:
fixes *p* :: *'a* :: *semiring-1 poly*
shows *mpoly-of-poly i* (*pCons c p*) = *Const c* + *Var i* $*$ *mpoly-of-poly i p*
⟨*proof*⟩

lemma *mpoly-of-poly-1* [*simp*]: *mpoly-of-poly i 1 = 1*
⟨*proof*⟩

lemma *mpoly-of-poly-uminus* [*simp*]: *mpoly-of-poly i* (−*p*) = −*mpoly-of-poly i p*
⟨*proof*⟩

lemma *mpoly-of-poly-add* [*simp*]: *mpoly-of-poly i* (*p* + *q*) = *mpoly-of-poly i p* + *mpoly-of-poly i q*
⟨*proof*⟩

lemma *mpoly-of-poly-diff* [*simp*]: *mpoly-of-poly i* (*p* − *q*) = *mpoly-of-poly i p* − *mpoly-of-poly i q*
⟨*proof*⟩

lemma *mpoly-of-poly-smult* [*simp*]:
mpoly-of-poly i (*Polynomial.smult c p*) = *smult c* (*mpoly-of-poly i p*)
⟨*proof*⟩

lemma *mpoly-of-poly-mult* [*simp*]:
fixes *p q* :: *'a* :: *comm-semiring-1 poly*
shows *mpoly-of-poly i* (*p* ∗ *q*) = *mpoly-of-poly i p* ∗ *mpoly-of-poly i q*
⟨*proof*⟩

lemma *insertion-mpoly-of-poly* [*simp*]: *insertion f* (*mpoly-of-poly i p*) = *poly p* (*f i*)
⟨*proof*⟩

lemma *mapping-of-mpoly-of-poly* [*simp*]: *mapping-of* (*mpoly-of-poly i p*) = *mpoly-of-poly-aux i p*
⟨*proof*⟩

lemma *vars-mpoly-of-poly*: *vars* (*mpoly-of-poly i p*) ⊆ {*i*}
⟨*proof*⟩

lemma *mpoly-map-vars-mpoly-of-poly* [*simp*]:
assumes *bij f*
shows *mpoly-map-vars f* (*mpoly-of-poly i p*) = *mpoly-of-poly* (*f i*) *p*
⟨*proof*⟩

end

5 More facts about algebraic numbers

theory *More-Algebraic-Numbers-HLW*
imports *Algebraic-Numbers.Algebraic-Numbers*
begin

5.1 Miscellaneous

lemma *in-Ints-imp-algebraic* [*simp*, *intro*]: $x \in \mathbb{Z} \Longrightarrow$ *algebraic* x
⟨*proof*⟩

lemma *in-Rats-imp-algebraic* [*simp*, *intro*]: $x \in \mathbb{Q} \Longrightarrow$ *algebraic* x
⟨*proof*⟩

lemma *algebraic-uminus-iff* [*simp*]: *algebraic* $(-x) \longleftrightarrow$ *algebraic* x
⟨*proof*⟩

lemma *algebraic-0* [*simp*]: *algebraic* $(0 :: 'a ::$ *field-char-0*$)$
and *algebraic-1* [*simp*]: *algebraic* $(1 :: 'a ::$ *field-char-0*$)$
⟨*proof*⟩

lemma *algebraic-sum* [*intro*]:
$(\bigwedge x.\ x \in A \Longrightarrow$ *algebraic* $(f\ x)) \Longrightarrow$ *algebraic* $(sum\ f\ A)$
⟨*proof*⟩

lemma *algebraic-prod* [*intro*]:
$(\bigwedge x.\ x \in A \Longrightarrow$ *algebraic* $(f\ x)) \Longrightarrow$ *algebraic* $(prod\ f\ A)$
⟨*proof*⟩

lemma *algebraic-sum-list* [*intro*]:
$(\bigwedge x.\ x \in set\ xs \Longrightarrow$ *algebraic* $x) \Longrightarrow$ *algebraic* $(sum\text{-}list\ xs)$
⟨*proof*⟩

lemma *algebraic-prod-list* [*intro*]:
$(\bigwedge x.\ x \in set\ xs \Longrightarrow$ *algebraic* $x) \Longrightarrow$ *algebraic* $(prod\text{-}list\ xs)$
⟨*proof*⟩

lemma *algebraic-sum-mset* [*intro*]:
$(\bigwedge x.\ x \in\# A \Longrightarrow$ *algebraic* $x) \Longrightarrow$ *algebraic* $(sum\text{-}mset\ A)$
⟨*proof*⟩

lemma *algebraic-prod-mset* [*intro*]:
$(\bigwedge x.\ x \in\# A \Longrightarrow$ *algebraic* $x) \Longrightarrow$ *algebraic* $(prod\text{-}mset\ A)$
⟨*proof*⟩

lemma *algebraic-power* [*intro*]: *algebraic* $x \Longrightarrow$ *algebraic* $(x \mathbin{\hat{}} n)$
⟨*proof*⟩

lemma *algebraic-csqrt* [*intro*]: *algebraic* $x \Longrightarrow$ *algebraic* $(csqrt\ x)$
⟨*proof*⟩

lemma *algebraic-csqrt-iff* [*simp*]: *algebraic* $(csqrt\ x) \longleftrightarrow$ *algebraic* x
⟨*proof*⟩

lemmas [*intro*] = *algebraic-plus algebraic-times algebraic-uminus algebraic-div*

lemma *algebraic-power-iff* [*simp*]:
assumes $n > 0$
shows *algebraic* $(x \mathbin{\hat{}} n) \longleftrightarrow$ *algebraic* x
⟨*proof*⟩

lemma *algebraic-ii* [*simp*]: *algebraic* i
⟨*proof*⟩

lemma *algebraic-int-fact* [*simp*, *intro*]: *algebraic-int* (*fact* n)
⟨*proof*⟩

lemma *algebraic-minus* [*intro*]: *algebraic* $x \Longrightarrow$ *algebraic* $y \Longrightarrow$ *algebraic* $(x - y)$
⟨*proof*⟩

lemma *algebraic-add-cancel-left* [*simp*]:
assumes *algebraic* x
shows *algebraic* $(x + y) \longleftrightarrow$ *algebraic* y
⟨*proof*⟩

lemma *algebraic-add-cancel-right* [*simp*]:
assumes *algebraic* y
shows *algebraic* $(x + y) \longleftrightarrow$ *algebraic* x
⟨*proof*⟩

lemma *algebraic-diff-cancel-left* [*simp*]:
assumes *algebraic* x
shows *algebraic* $(x - y) \longleftrightarrow$ *algebraic* y
⟨*proof*⟩

lemma *algebraic-diff-cancel-right* [*simp*]:
assumes *algebraic* y
shows *algebraic* $(x - y) \longleftrightarrow$ *algebraic* x
⟨*proof*⟩

lemma *algebraic-mult-cancel-left* [*simp*]:
assumes *algebraic* x $x \neq 0$
shows *algebraic* $(x * y) \longleftrightarrow$ *algebraic* y
⟨*proof*⟩

lemma *algebraic-mult-cancel-right* [*simp*]:
assumes *algebraic* y $y \neq 0$
shows *algebraic* $(x * y) \longleftrightarrow$ *algebraic* x
⟨*proof*⟩

lemma *algebraic-inverse-iff* [*simp*]: *algebraic* (*inverse* y) $\longleftrightarrow$ *algebraic* y
⟨*proof*⟩

lemma *algebraic-divide-cancel-left* [*simp*]:
assumes *algebraic* x $x \neq 0$

shows *algebraic* $(x\ /\ y) \longleftrightarrow$ *algebraic* y
⟨*proof*⟩

lemma *algebraic-divide-cancel-right* [*simp*]:
assumes *algebraic* y $y \neq 0$
shows *algebraic* $(x\ /\ y) \longleftrightarrow$ *algebraic* x
⟨*proof*⟩

5.2 Turning an algebraic number into an algebraic integer

5.3 Multiplying an algebraic number with a suitable integer turns it into an algebraic integer.

lemma *algebraic-imp-algebraic-int*:
fixes $x :: 'a :: \textit{field-char-0}$
assumes *ipoly* p $x = 0$ $p \neq 0$
defines $c \equiv \textit{Polynomial.lead-coeff}\ p$
shows *algebraic-int* $(\textit{of-int}\ c * x)$
⟨*proof*⟩

lemma *algebraic-imp-algebraic-int'*:
fixes $x :: 'a :: \textit{field-char-0}$
assumes *ipoly* p $x = 0$ $p \neq 0$ *Polynomial.lead-coeff* p *dvd* c
shows *algebraic-int* $(\textit{of-int}\ c * x)$
⟨*proof*⟩

end

6 Miscellaneous facts

theory *Misc-HLW*
imports
Complex-Main
HOL−Library.Multiset
HOL−Library.FuncSet
HOL−Library.Groups-Big-Fun
HOL−Library.Poly-Mapping
HOL−Library.Landau-Symbols
HOL−Combinatorics.Permutations
HOL−Computational-Algebra.Computational-Algebra
begin

lemma *set-mset-subset-singletonD*:
assumes *set-mset* $A \subseteq \{x\}$
shows $A =$ *replicate-mset* $(\textit{size}\ A)\ x$
⟨*proof*⟩

lemma *image-mset-eq-replicate-msetD*:
assumes *image-mset* $f\ A =$ *replicate-mset* $n\ y$

shows $\forall x{\in}\#A.\ f\ x = y$
⟨*proof*⟩

lemma *bij-betw-permutes-compose-left*:
assumes π *permutes* A
shows *bij-betw* $(\lambda\sigma.\ \pi \circ \sigma)$ $\{\sigma.\ \sigma$ *permutes* $A\}$ $\{\sigma.\ \sigma$ *permutes* $A\}$
⟨*proof*⟩

lemma *bij-betw-compose-left-perm-Pi*:
assumes π *permutes* B
shows *bij-betw* $(\lambda f.\ (\pi \circ f))$ $(A \to B)$ $(A \to B)$
⟨*proof*⟩

lemma *bij-betw-compose-left-perm-PiE*:
assumes π *permutes* B
shows *bij-betw* $(\lambda f.$ *restrict* $(\pi \circ f)$ $A)$ $(A \to_E B)$ $(A \to_E B)$
⟨*proof*⟩

lemma *bij-betw-image-mset-set*:
assumes *bij-betw f A B*
shows *image-mset f* (*mset-set A*) = *mset-set B*
⟨*proof*⟩

lemma *finite-multisets-of-size*:
assumes *finite A*
shows *finite* $\{X.$ *set-mset* $X \subseteq A \wedge$ *size* $X = n\}$
⟨*proof*⟩

lemma *sum-mset-image-mset-sum-mset-image-mset*:
sum-mset (*image-mset g* (*sum-mset* (*image-mset f A*))) =
sum-mset (*image-mset* $(\lambda x.$ *sum-mset* (*image-mset g* (*f x*))) *A*)
⟨*proof*⟩

lemma *sum-mset-image-mset-singleton*: *sum-mset* (*image-mset* $(\lambda x.\ \{\#f\ x\#\})$ *A*)
= *image-mset f A*
⟨*proof*⟩

lemma *sum-mset-conv-sum*:
sum-mset (*image-mset f A*) = $(\sum x{\in}$*set-mset A*. *of-nat* (*count A x*) $*$ *f x*)
⟨*proof*⟩

lemma *sum-mset-conv-Sum-any*:
sum-mset (*image-mset f A*) = *Sum-any* $(\lambda x.$ *of-nat* (*count A x*) $*$ *f x*)
⟨*proof*⟩

lemma *Sum-any-sum-swap*:
assumes *finite A* $\bigwedge y.$ *finite* $\{x.\ f\ x\ y \neq 0\}$
shows *Sum-any* $(\lambda x.$ *sum* (*f x*) *A*) = $(\sum y{\in}A.$ *Sum-any* $(\lambda x.\ f\ x\ y))$
⟨*proof*⟩

lemma (**in** *landau-pair*) *big-power*:
assumes $f \in L\ F\ g$
shows $(\lambda x.\ f\ x\ \widehat{}\ n) \in L\ F\ (\lambda x.\ g\ x\ \widehat{}\ n)$
⟨*proof*⟩

lemma (**in** *landau-pair*) *small-power*:
assumes $f \in l\ F\ g\ n > 0$
shows $(\lambda x.\ f\ x\ \widehat{}\ n) \in l\ F\ (\lambda x.\ g\ x\ \widehat{}\ n)$
⟨*proof*⟩

lemma *pairwise-imp-disjoint-family-on*:
assumes *pairwise R A*
assumes $\bigwedge m\ n.\ m \in A \Longrightarrow n \in A \Longrightarrow R\ m\ n \Longrightarrow f\ m \cap f\ n = \{\}$
shows *disjoint-family-on f A*
⟨*proof*⟩

lemma (**in** *comm-monoid-set*) *If-eq*:
assumes $y \in A$ *finite A*
shows *F* (λ*x*. *g x* (*if x* = *y then h1 x else h2 x*)) *A* = *f* (*g y* (*h1 y*)) (*F* (λ*x*. *g x* (*h2 x*)) (*A*−{*y*}))
⟨*proof*⟩

lemma *prod-nonzeroI*:
fixes *f* :: *'a* ⇒ *'b* :: {*semiring-no-zero-divisors*, *comm-semiring-1*}
assumes $\bigwedge x.\ x \in A \Longrightarrow f\ x \neq 0$
shows *prod f A* ≠ *0*
⟨*proof*⟩

lemma *frequently-prime-cofinite*: *frequently* (*prime* :: *nat* ⇒ *bool*) *cofinite*
⟨*proof*⟩

lemma *frequently-eventually-mono*:
assumes *frequently Q F eventually P F* $\bigwedge x.\ P\ x \Longrightarrow Q\ x \Longrightarrow R\ x$
shows *frequently R F*
⟨*proof*⟩

lemma *bij-betw-Diff*:
assumes *bij-betw f A B bij-betw f A' B' A'* ⊆ *A B'* ⊆ *B*
shows *bij-betw f* (*A* − *A'*) (*B* − *B'*)
⟨*proof*⟩

lemma *bij-betw-singleton*: *bij-betw f* {*x*} {*y*} ⟷ *f x* = *y*
⟨*proof*⟩

end

7 The Hermite–Lindemann–WeierstraßTranscendence Theorem

theory *Hermite-Lindemann*
imports
Pi-Transcendental.Pi-Transcendental
Algebraic-Numbers.Algebraic-Numbers
Algebraic-Integer-Divisibility
More-Min-Int-Poly
Complex-Lexorder
More-Polynomial-HLW
More-Multivariate-Polynomial-HLW
More-Algebraic-Numbers-HLW
Misc-HLW
begin

hide-const (**open**) *Henstock-Kurzweil-Integration.content Module.smult*

The Hermite–Lindemann–Weierstraßtheorem answers questions about the transcendence of the exponential function and other related complex functions. It proves that a large number of combinations of exponentials is always transcendental.

A first (much weaker) version of the theorem was proven by Hermite. Lindemann and Weierstraßthen successively generalised it shortly afterwards, and finally Baker gave another, arguably more elegant formulation (which is the one that we will prove, and then derive the traditional version from it).

To honour the contributions of all three of these 19th-century mathematicians, I refer to the theorem as the Hermite–Lindemann–Weierstraßtheorem, even though in other literature it is often called Hermite–Lindemann or Lindemann–Weierstraß. To keep things short, the Isabelle name of the theorem, however, will omit Weierstraß's name.

7.1 Main proof

Following Baker, We first prove the following special form of the theorem: Let $m > 0$ and $q_1, \ldots, q_m \in \mathbb{Z}[X]$ be irreducible, non-constant, and pairwise coprime polynomials. Let $\beta_1, \ldots, \beta_m$ be non-zero integers. Then

$$\sum_{i=1}^{m} \beta_i \sum_{q_i(\alpha)=0} e^{\alpha} \neq 0$$

The difference to the final theorem is that

1. The coefficients β_i are non-zero integers (as opposed to arbitrary algebraic numbers)

2. The exponents α_i occur in full sets of conjugates, and each set has the same coefficient.

In a similar fashion to the proofs of the transcendence of e and π, we define some number J depending on the α_i and β_i and an arbitrary sufficiently large prime p. We then show that, on one hand, J is an integer multiple of $(p-1)!$, but on the other hand it is bounded from above by a term of the form $C_1 \cdot C_2^p$. This is then clearly a contradiction if p is chosen large enough.

lemma *Hermite-Lindemann-aux1*:
fixes P :: *int poly set* **and** β :: *int poly* $\Rightarrow$ *int*
assumes *finite* P **and** $P \neq \{\}$
assumes *distinct*: *pairwise Rings.coprime P*
assumes *irred*: $\bigwedge p.\ p \in P \Longrightarrow$ *irreducible p*
assumes *nonconstant*: $\bigwedge p.\ p \in P \Longrightarrow$ *Polynomial.degree* $p > 0$
assumes β-*nz*: $\bigwedge p.\ p \in P \Longrightarrow \beta\ p \neq 0$
defines *Roots* $\equiv (\lambda p.\ \{\alpha$::*complex. poly* (*of-int-poly p*) $\alpha = 0\})$
shows $(\sum p{\in}P.$ *of-int* $(\beta\ p) * (\sum \alpha{\in}$*Roots p. exp* $\alpha)) \neq 0$
⟨*proof*⟩

7.2 Removing the restriction of full sets of conjugates

We will now remove the restriction that the α_i must occur in full sets of conjugates by multiplying the equality with all permutations of roots.

lemma *Hermite-Lindemann-aux2*:
fixes X :: *complex set* **and** β :: *complex* $\Rightarrow$ *int*
assumes *finite X*
assumes *nz*: $\bigwedge x.\ x \in X \Longrightarrow \beta\ x \neq 0$
assumes *alg*: $\bigwedge x.\ x \in X \Longrightarrow$ *algebraic x*
assumes *sum0*: $(\sum x{\in}X.$ *of-int* $(\beta\ x) *$ *exp* $x) = 0$
shows $X = \{\}$
⟨*proof*⟩

7.3 Removing the restriction to integer coefficients

Next, we weaken the restriction that the β_i must be integers to the restriction that they must be rationals. This is done simply by multiplying with the least common multiple of the demoninators.

lemma *Hermite-Lindemann-aux3*:
fixes X :: *complex set* **and** β :: *complex* $\Rightarrow$ *rat*
assumes *finite X*
assumes *nz*: $\bigwedge x.\ x \in X \Longrightarrow \beta\ x \neq 0$
assumes *alg*: $\bigwedge x.\ x \in X \Longrightarrow$ *algebraic x*
assumes *sum0*: $(\sum x{\in}X.$ *of-rat* $(\beta\ x) *$ *exp* $x) = 0$
shows $X = \{\}$
⟨*proof*⟩

Next, we weaken the restriction that the β_i must be rational to them being algebraic. Similarly to before, this is done by multiplying over all possible permutations of the β_i (in some sense) to introduce more symmetry, from which it then follows by the fundamental theorem of symmetric polynomials that the resulting coefficients are rational.

lemma *Hermite-Lindemann-aux4*:
  fixes β :: *complex* $\Rightarrow$ *complex*
  assumes [*intro*]: *finite X*
  assumes *alg1*: $\bigwedge x.\ x \in X \Longrightarrow$ *algebraic x*
  assumes *alg2*: $\bigwedge x.\ x \in X \Longrightarrow$ *algebraic* (β *x*)
  assumes *nz*: $\bigwedge x.\ x \in X \Longrightarrow \beta\ x \neq 0$
  assumes *sum0*: $(\sum x{\in}X.\ \beta\ x * \mathit{exp}\ x) = 0$
  shows $X = \{\}$
⟨*proof*⟩

7.4 The final theorem

We now additionally allow some of the β_i to be zero:

lemma *Hermite-Lindemann′*:
  fixes β :: *complex* $\Rightarrow$ *complex*
  assumes *finite X*
  assumes $\bigwedge x.\ x \in X \Longrightarrow$ *algebraic x*
  assumes $\bigwedge x.\ x \in X \Longrightarrow$ *algebraic* (β *x*)
  assumes $(\sum x{\in}X.\ \beta\ x * \mathit{exp}\ x) = 0$
  shows $\forall x{\in}X.\ \beta\ x = 0$
⟨*proof*⟩

Lastly, we switch to indexed summation in order to obtain a version of the theorem that is somewhat nicer to use:

theorem *Hermite-Lindemann*:
  fixes $\alpha\ \beta$ :: *'a* $\Rightarrow$ *complex*
  assumes *finite I*
  assumes $\bigwedge x.\ x \in I \Longrightarrow$ *algebraic* (α *x*)
  assumes $\bigwedge x.\ x \in I \Longrightarrow$ *algebraic* (β *x*)
  assumes *inj-on* α *I*
  assumes $(\sum x{\in}I.\ \beta\ x * \mathit{exp}\ (\alpha\ x)) = 0$
  shows $\forall x{\in}I.\ \beta\ x = 0$
⟨*proof*⟩

The following version using lists instead of sequences is even more convenient to use in practice:

corollary *Hermite-Lindemann-list*:
  fixes *xs* :: (*complex* × *complex*) *list*
  assumes *alg*: $\forall (x,y){\in}\mathit{set}\ \mathit{xs}.\ \mathit{algebraic}\ x \wedge \mathit{algebraic}\ y$
  assumes *distinct*: *distinct* (*map snd xs*)
  assumes *sum0*: $(\sum (c,\alpha){\leftarrow}\mathit{xs}.\ c * \mathit{exp}\ \alpha) = 0$
  shows $\forall c{\in}(\mathit{fst}\ `\ \mathit{set}\ \mathit{xs}).\ c = 0$
⟨*proof*⟩

7.5 The traditional formulation of the theorem

What we proved above was actually Baker's reformulation of the theorem. Thus, we now also derive the original one, which uses linear independence and algebraic independence.

It states that if $\alpha_1, \ldots, \alpha_n$ are algebraic numbers that are linearly independent over $\mathbb{Z}$, then $e^{\alpha_1}, \ldots, e^{\alpha_n}$ are algebraically independent over $\mathbb{Q}$.

Linear independence over the integers is just independence of a set of complex numbers when viewing the complex numbers as a $\mathbb{Z}$-module.

definition *linearly-independent-over-int* :: *'a* :: *field-char-0 set* ⇒ *bool* **where**
linearly-independent-over-int = *module.independent* (λ*r x*. *of-int r* ∗ *x*)

Algebraic independence over the rationals means that the given set X of numbers fulfils no non-trivial polynomial equation with rational coefficients, i.e. there is no non-zero multivariate polynomial with rational coefficients that, when inserting the numbers from X, becomes zero.

Note that we could easily replace 'rational coefficients' with 'algebraic coefficients' here and the proof would still go through without any modifications.

definition *algebraically-independent-over-rat* :: *nat* ⇒ (*nat* ⇒ *'a* :: *field-char-0*) ⇒ *bool* **where**
algebraically-independent-over-rat n a ⟷
(∀ *p*. *vars p* ⊆ {..<*n*} ∧ (∀ *m*. *coeff p m* ∈ ℚ) ∧ *insertion a p* = *0* ⟶ *p* = *0*)

corollary *Hermite-Lindemann-original*:
fixes *n* :: *nat* **and** α :: *nat* ⇒ *complex*
assumes *inj-on* α {..<*n*}
assumes ⋀*i*. *i* < *n* ⟹ *algebraic* (α *i*)
assumes *linearly-independent-over-int* (α ' {..<*n*})
shows *algebraically-independent-over-rat n* (λ*i*. *exp* (α *i*))
⟨*proof*⟩

7.6 Simple corollaries

Now, we derive all the usual obvious corollaries of the theorem in the obvious way.

First, the exponential of a non-zero algebraic number is transcendental.

corollary *algebraic-exp-complex-iff*:
assumes *algebraic x*
shows *algebraic* (*exp x* :: *complex*) ⟷ *x* = *0*
⟨*proof*⟩

More generally, any sum of exponentials with algebraic coefficients and exponents is transcendental if the exponents are all distinct and non-zero and at least one coefficient is non-zero.

corollary *sum-of-exp-transcendentalI*:

fixes *xs* :: (*complex* × *complex*) *list*
assumes ∀(*x*,*y*)∈*set xs*. *algebraic x* ∧ *algebraic y* ∧ *y* ≠ *0*
assumes ∃*x*∈*fst*‘*set xs*. *x* ≠ *0*
assumes *distinct*: *distinct* (*map snd xs*)
shows ¬*algebraic* (∑(*c*,*α*)←*xs*. *c* ∗ *exp α*)
⟨*proof*⟩

Any complex logarithm of an algebraic number other than 1 is transcendental (no matter which branch cut).

corollary *transcendental-complex-logarithm*:
assumes *algebraic x exp y* = (*x* :: *complex*) *x* ≠ *1*
shows ¬*algebraic y*
⟨*proof*⟩

In particular, this holds for the standard branch of the logarithm.

corollary *transcendental-Ln*:
assumes *algebraic x x* ≠ *0 x* ≠ *1*
shows ¬*algebraic* (*Ln x*)
⟨*proof*⟩

The transcendence of e and π, which I have already formalised directly in other AFP entries, now follows as a simple corollary.

corollary *exp-1-complex-transcendental*: ¬*algebraic* (*exp 1* :: *complex*)
⟨*proof*⟩

corollary *pi-transcendental*: ¬*algebraic pi*
⟨*proof*⟩

7.7 Transcendence of the trigonometric and hyperbolic functions

In a similar fashion, we can also prove the transcendence of all the trigonometric and hyperbolic functions such as sin, tan, sinh, arcsin, etc.

lemma *transcendental-sinh*:
assumes *algebraic z z* ≠ *0*
shows ¬*algebraic* (*sinh z* :: *complex*)
⟨*proof*⟩

lemma *transcendental-cosh*:
assumes *algebraic z z* ≠ *0*
shows ¬*algebraic* (*cosh z* :: *complex*)
⟨*proof*⟩

lemma *transcendental-sin*:
assumes *algebraic z z* ≠ *0*
shows ¬*algebraic* (*sin z* :: *complex*)
⟨*proof*⟩

lemma *transcendental-cos*:
assumes *algebraic z z* ≠ *0*
shows ¬*algebraic* (*cos z* :: *complex*)
⟨*proof*⟩

lemma *tan-square-neq-neg1*: *tan* (*z* :: *complex*) ^ *2* ≠ −*1*
⟨*proof*⟩

lemma *transcendental-tan*:
assumes *algebraic z z* ≠ *0*
shows ¬*algebraic* (*tan z* :: *complex*)
⟨*proof*⟩

lemma *transcendental-cot*:
assumes *algebraic z z* ≠ *0*
shows ¬*algebraic* (*cot z* :: *complex*)
⟨*proof*⟩

lemma *transcendental-tanh*:
assumes *algebraic z z* ≠ *0 cosh z* ≠ *0*
shows ¬*algebraic* (*tanh z* :: *complex*)
⟨*proof*⟩

lemma *transcendental-Arcsin*:
assumes *algebraic z z* ≠ *0*
shows ¬*algebraic* (*Arcsin z*)
⟨*proof*⟩

lemma *transcendental-Arccos*:
assumes *algebraic z z* ≠ *1*
shows ¬*algebraic* (*Arccos z*)
⟨*proof*⟩

lemma *transcendental-Arctan*:
assumes *algebraic z z* ∉ {*0*, i, −i}
shows ¬*algebraic* (*Arctan z*)
⟨*proof*⟩

end

References

[1] A. Baker. *Transcendental Number Theory*. Cambridge Mathematical Library. Cambridge University Press, 1975.

[2] S. Bernard. Formalization of the Lindemann-Weierstrass theorem. In

M. Ayala-Rincón and C. A. Muñoz, editors, *Interactive Theorem Proving - 8th International Conference, ITP 2017, Brasília, Brazil, September 26-29, 2017, Proceedings*, volume 10499 of *Lecture Notes in Computer Science*, pages 65–80. Springer, 2017.

[3] R. Steinberg and R. M. Redheffer. Analytic proof of the Lindemann theorem. *Pacific Journal of Mathematics*, 2(2):231–242, 1952.