The Irrationality of $\zeta(3)$

Manuel Eberl

June 20, 2024

Abstract

This article provides a formalisation of Beukers's straightforward analytic proof [2] that $\zeta(3)$ is irrational. This was first proven by Apéry [1] (which is why this result is also often called 'Apéry's Theorem') using a more algebraic approach. This formalisation follows Filaseta's presentation of Beukers's proof [5].

Contents

1 The Irrationality of $\zeta(3)$

theory *Zeta-3-Irrational*
imports
E-Transcendental.E-Transcendental
Prime-Number-Theorem.Prime-Number-Theorem
Prime-Distribution-Elementary.PNT-Consequences
begin

hide-const (**open**) *UnivPoly.coeff* *UnivPoly.up-ring.monom*
hide-const (**open**) *Module.smult* *Coset.order*

Apéry's original proof of the irrationality of $\zeta(3)$ contained several tricky identities of sums involving binomial coefficients that are difficult to prove. I instead follow Beukers's proof, which consists of several fairly straightforward manipulations of integrals with absolutely no caveats.

Both Apéry and Beukers make use of an asymptotic upper bound on $\text{lcm}\{1\ldots n\}$ – namely $\text{lcm}\{1\ldots n\} \in o(c^n)$ for any $c > e$, which is a consequence of the Prime Number Theorem (which, fortunately, is available in the *Archive of Formal Proofs* [4, 3]).

I follow the concise presentation of Beukers's proof in Filaseta's lecture notes [5], which turned out to be very amenable to formalisation.

There is another earlier formalisation of the irrationality of $\zeta(3)$ by Mahboubi *et al.* [6], who followed Apéry's original proof, but were ultimately forced to find a more elementary way to prove the asymptotics of $\text{lcm}\{1\ldots n\}$ than the Prime Number Theorem.

First, we will require some auxiliary material before we get started with the actual proof.

1.1 Auxiliary facts about polynomials

lemma *higher-pderiv-minus*: $(pderiv\ \widehat{}\widehat{}\ n)\ (-p :: 'a :: idom\ poly) = -(pderiv\ \widehat{}\widehat{}\ n)\ p$
by (*induction n*) (*auto simp*: *pderiv-minus*)

lemma *pderiv-power*: $pderiv\ (p\ \widehat{}\ n) = smult\ (of\text{-}nat\ n)\ (p\ \widehat{}\ (n - 1)) * pderiv\ p$
by (*cases n*) (*simp-all add*: *pderiv-power-Suc del*: *power-Suc*)

lemma *higher-pderiv-monom*:
$k \le n \Longrightarrow (pderiv\ \widehat{}\widehat{}\ k)\ (monom\ c\ n) = monom\ (of\text{-}nat\ (pochhammer\ (n - k + 1)\ k) * c)\ (n - k)$
by (*induction k*) (*auto simp*: *pderiv-monom pochhammer-rec Suc-diff-le Suc-diff-Suc mult-ac*)

lemma *higher-pderiv-mult*:
$(pderiv\ \widehat{}\widehat{}\ n)\ (p * q) =$

(∑ *k≤n. Polynomial.smult (of-nat (n choose k)) ((pderiv ^^ k) p ∗ (pderiv ^^ (n − k)) q))*
proof (*induction n*)
case (*Suc n*)
have *eq*: (*Suc n choose k*) = (*n choose k*) + (*n choose* (*k−1*)) **if** *k > 0* **for** *k*
using *that* **by** (*cases k*) *auto*
have (*pderiv ^^ Suc n*) (*p ∗ q*) =
(∑ *k≤n. smult* (*of-nat* (*n choose k*)) ((*pderiv ^^ k*) *p ∗* (*pderiv ^^* (*Suc n − k*)) *q*)) +
(∑ *k≤n. smult* (*of-nat* (*n choose k*)) ((*pderiv ^^ Suc k*) *p ∗* (*pderiv ^^* (*n − k*)) *q*))
by (*simp add*: *Suc pderiv-sum pderiv-smult pderiv-mult sum.distrib smult-add-right algebra-simps Suc-diff-le*)
also have (∑ *k≤n. smult* (*of-nat* (*n choose k*)) ((*pderiv ^^ k*) *p ∗* (*pderiv ^^* (*Suc n − k*)) *q*)) =
(∑ *k∈insert 0* {*1..n*}. *smult* (*of-nat* (*n choose k*)) ((*pderiv ^^ k*) *p ∗* (*pderiv ^^* (*Suc n − k*)) *q*))
by (*intro sum.cong*) *auto*
also have . . . = (∑ *k=1..n. smult* (*of-nat* (*n choose k*)) ((*pderiv ^^ k*) *p ∗* (*pderiv ^^* (*Suc n − k*)) *q*)) + *p ∗* (*pderiv ^^ Suc n*) *q*
by (*subst sum.insert*) (*auto simp*: *add-ac*)
also have (∑ *k≤n. smult* (*of-nat* (*n choose k*)) ((*pderiv ^^ Suc k*) *p ∗* (*pderiv ^^* (*n − k*)) *q*)) =
(∑ *k=1..n+1. smult* (*of-nat* (*n choose* (*k−1*))) ((*pderiv ^^ k*) *p ∗* (*pderiv ^^* (*Suc n − k*)) *q*))
by (*intro sum.reindex-bij-witness*[*of - λk. k − 1 Suc*]) *auto*
also have . . . = (∑ *k∈insert* (*n+1*) {*1..n*}. *smult* (*of-nat* (*n choose* (*k−1*))) ((*pderiv ^^ k*) *p ∗* (*pderiv ^^* (*Suc n − k*)) *q*))
by (*intro sum.cong*) *auto*
also have . . . = (∑ *k=1..n. smult* (*of-nat* (*n choose* (*k−1*))) ((*pderiv ^^ k*) *p ∗* (*pderiv ^^* (*Suc n − k*)) *q*)) + (*pderiv ^^ Suc n*) *p ∗ q*
by (*subst sum.insert*) (*auto simp*: *add-ac*)
also have (∑ *k=1..n. smult* (*of-nat* (*n choose k*)) ((*pderiv ^^ k*) *p ∗* (*pderiv ^^* (*Suc n − k*)) *q*)) +
p ∗ (*pderiv ^^ Suc n*) *q* + . . . =
(∑ *k=1..n. smult* (*of-nat* (*Suc n choose k*)) ((*pderiv ^^ k*) *p ∗* (*pderiv ^^* (*Suc n − k*)) *q*)) +
p ∗ (*pderiv ^^ Suc n*) *q* + (*pderiv ^^ Suc n*) *p ∗ q*
by (*simp add*: *sum.distrib algebra-simps smult-add-right eq smult-add-left*)
also have . . . = (∑ *k∈*{*1..n*}∪{*0,Suc n*}. *smult* (*of-nat* (*Suc n choose k*)) ((*pderiv ^^ k*) *p ∗* (*pderiv ^^* (*Suc n − k*)) *q*))
by (*subst sum.union-disjoint*) (*auto simp*: *algebra-simps*)
also have {*1..n*}∪{*0,Suc n*} = {*..Suc n*} **by** *auto*
finally show *?case* **.**
qed *auto*

1.2 Auxiliary facts about integrals

theorem (**in** *pair-sigma-finite*) *Fubini-set-integrable*:

```
  fixes f :: - ⇒ -::{banach, second-countable-topology}
  assumes f[measurable]: set-borel-measurable (M1 ⨂M M2) (A × B) f
    and integ1: set-integrable M1 A (λx. ∫ y∈B. norm (f (x, y)) ∂M2)
    and integ2: AE x∈A in M1. set-integrable M2 B (λy. f (x, y))
  shows set-integrable (M1 ⨂M M2) (A × B) f
  unfolding set-integrable-def
proof (rule Fubini-integrable)
  note integ1
  also have set-integrable M1 A (λx. ∫ y∈B. norm (f (x, y)) ∂M2) ⟷
      integrable M1 (λx. LINT y|M2. norm (indicat-real (A × B) (x, y) *R f (x,
y)))
    unfolding set-integrable-def
   by (intro Bochner-Integration.integrable-cong) (auto simp: indicator-def set-lebesgue-integral-def)
  finally show . . . .
next
  from integ2 show AE x in M1. integrable M2 (λy. indicat-real (A × B) (x, y)
*R f (x, y))
  proof eventually-elim
    case (elim x)
    show integrable M2 (λy. indicat-real (A × B) (x, y) *R f (x, y))
    proof (cases x ∈ A)
      case True
      with elim have set-integrable M2 B (λy. f (x, y)) by simp
      also have ?this ⟷ ?thesis
        unfolding set-integrable-def using True
        by (intro Bochner-Integration.integrable-cong) (auto simp: indicator-def)
      finally show ?thesis .
    qed auto
  qed
qed (insert assms, auto simp: set-borel-measurable-def)

lemma (in pair-sigma-finite) set-integral-fst′:
  fixes f :: - ⇒ ′c :: {second-countable-topology, banach}
  assumes set-integrable (M1 ⨂M M2) (A × B) f
  shows   set-lebesgue-integral (M1 ⨂M M2) (A × B) f =
          (∫ x∈A. (∫ y∈B. f (x, y) ∂M2) ∂M1)
proof −
  have set-lebesgue-integral (M1 ⨂M M2) (A × B) f =
        (∫ z. indicator (A × B) z *R f z ∂(M1 ⨂M M2))
    by (simp add: set-lebesgue-integral-def)
  also have . . . = (∫ x. ∫ y. indicator (A × B) (x,y) *R f (x,y) ∂M2 ∂M1)
    using assms by (subst integral-fst′ [symmetric]) (auto simp: set-integrable-def)
  also have . . . = (∫ x∈A. (∫ y∈B. f (x,y) ∂M2) ∂M1)
    unfolding set-lebesgue-integral-def
    by (intro Bochner-Integration.integral-cong refl) (auto simp: indicator-def)
  finally show ?thesis .
qed

lemma (in pair-sigma-finite) set-integral-snd:
```

fixes *f* :: - ⇒ *'c* :: {*second-countable-topology*, *banach*}
assumes *set-integrable* (*M1* $\bigotimes_M$ *M2*) (*A* × *B*) *f*
shows *set-lebesgue-integral* (*M1* $\bigotimes_M$ *M2*) (*A* × *B*) *f* =
(∫ *y*∈*B*. (∫ *x*∈*A*. *f* (*x*, *y*) ∂*M1*) ∂*M2*)
proof −
have *set-lebesgue-integral* (*M1* $\bigotimes_M$ *M2*) (*A* × *B*) *f* =
(∫ *z*. *indicator* (*A* × *B*) *z* $*_R$ *f z* ∂(*M1* $\bigotimes_M$ *M2*))
by (*simp add*: *set-lebesgue-integral-def*)
also have ... = (∫ *y*. ∫ *x*. *indicator* (*A* × *B*) (*x*,*y*) $*_R$ *f* (*x*,*y*) ∂*M1* ∂*M2*)
using *assms* **by** (*subst integral-snd*) (*auto simp*: *set-integrable-def case-prod-unfold*)
also have ... = (∫ *y*∈*B*. (∫ *x*∈*A*. *f* (*x*,*y*) ∂*M1*) ∂*M2*)
unfolding *set-lebesgue-integral-def*
by (*intro Bochner-Integration.integral-cong refl*) (*auto simp*: *indicator-def*)
finally show *?thesis* **.**
qed

proposition (**in** *pair-sigma-finite*) *Fubini-set-integral*:
fixes *f* :: - ⇒ - ⇒ - :: {*banach*, *second-countable-topology*}
assumes *f*: *set-integrable* (*M1* $\bigotimes_M$ *M2*) (*A* × *B*) (*case-prod f*)
shows (∫ *y*∈*B*. (∫ *x*∈*A*. *f x y* ∂*M1*) ∂*M2*) = (∫ *x*∈*A*. (∫ *y*∈*B*. *f x y* ∂*M2*) ∂*M1*)
proof −
have (∫ *y*∈*B*. (∫ *x*∈*A*. *f x y* ∂*M1*) ∂*M2*) = (∫ *y*. (∫ *x*. *indicator* (*A* × *B*) (*x*, *y*)
$*_R$ *f x y* ∂*M1*) ∂*M2*)
unfolding *set-lebesgue-integral-def*
by (*intro Bochner-Integration.integral-cong*) (*auto simp*: *indicator-def*)
also have ... = (∫ *x*. (∫ *y*. *indicator* (*A* × *B*) (*x*, *y*) $*_R$ *f x y* ∂*M2*) ∂*M1*)
using *assms* **by** (*intro Fubini-integral*) (*auto simp*: *set-integrable-def case-prod-unfold*)
also have ... = (∫ *x*∈*A*. (∫ *y*∈*B*. *f x y* ∂*M2*) ∂*M1*)
unfolding *set-lebesgue-integral-def*
by (*intro Bochner-Integration.integral-cong*) (*auto simp*: *indicator-def*)
finally show *?thesis* **.**
qed

lemma (**in** *pair-sigma-finite*) *nn-integral-swap*:
assumes [*measurable*]: *f* ∈ *borel-measurable* (*M1* $\bigotimes_M$ *M2*)
shows ($\int^+$*x*. *f x* ∂(*M1* $\bigotimes_M$ *M2*)) = ($\int^+$(*y*,*x*). *f* (*x*,*y*) ∂(*M2* $\bigotimes_M$ *M1*))
by (*subst distr-pair-swap*, *subst nn-integral-distr*) (*auto simp*: *case-prod-unfold*)

lemma *set-integrable-bound*:
fixes *f* :: *'a* ⇒ *'b*::{*banach*, *second-countable-topology*}
and *g* :: *'a* ⇒ *'c*::{*banach*, *second-countable-topology*}
shows *set-integrable M A f* ⟹ *set-borel-measurable M A g* ⟹
(*AE x in M*. *x* ∈ *A* ⟶ *norm* (*g x*) ≤ *norm* (*f x*)) ⟹ *set-integrable M*
A g
unfolding *set-integrable-def*
apply (*erule Bochner-Integration.integrable-bound*)
apply (*simp add*: *set-borel-measurable-def*)
apply (*erule eventually-mono*)
apply (*auto simp*: *indicator-def*)

done

lemma *set-integrableI-nonneg*:
fixes f :: $'a \Rightarrow real$
assumes *set-borel-measurable M A f*
assumes $AE\ x\ in\ M.\ x \in A \longrightarrow 0 \leq f\ x$ $(\int^+ x{\in}A.\ f\ x\ \partial M) < \infty$
shows *set-integrable M A f*
unfolding *set-integrable-def*
proof (*rule integrableI-nonneg*)
from *assms* **show** $(\lambda x.\ indicator\ A\ x *_R f\ x) \in borel\text{-}measurable\ M$
by (*simp add*: *set-borel-measurable-def*)
from *assms*(*2*) **show** $AE\ x\ in\ M.\ 0 \leq indicat\text{-}real\ A\ x *_R f\ x$
by *eventually-elim* (*auto simp*: *indicator-def*)
have $(\int^+ x.\ ennreal\ (indicator\ A\ x *_R f\ x)\ \partial M) = (\int^+ x{\in}A.\ f\ x\ \partial M)$
by (*intro nn-integral-cong*) (*auto simp*: *indicator-def*)
also have $\ldots < \infty$ **by** *fact*
finally show $(\int^+ x.\ ennreal\ (indicator\ A\ x *_R f\ x)\ \partial M) < \infty$ **.**
qed

lemma *set-integral-eq-nn-integral*:
assumes *set-borel-measurable M A f*
assumes $set\text{-}nn\text{-}integral\ M\ A\ f = ennreal\ x$ $x \geq 0$
assumes $AE\ x\ in\ M.\ x \in A \longrightarrow f\ x \geq 0$
shows *set-integrable M A f*
and $set\text{-}lebesgue\text{-}integral\ M\ A\ f = x$
proof −
have *eq*: $(\lambda x.\ (if\ x \in A\ then\ 1\ else\ 0) * f\ x) = (\lambda x.\ if\ x \in A\ then\ f\ x\ else\ 0)$
$(\lambda x.\ if\ x \in A\ then\ ennreal\ (f\ x)\ else\ 0) =$
$(\lambda x.\ ennreal\ (f\ x) * (if\ x \in A\ then\ 1\ else\ 0))$
$(\lambda x.\ ennreal\ (f\ x * (if\ x \in A\ then\ 1\ else\ 0))) =$
$(\lambda x.\ ennreal\ (f\ x) * (if\ x \in A\ then\ 1\ else\ 0))$
by *auto*
from *assms* **show** ∗: *set-integrable M A f*
by (*intro set-integrableI-nonneg*) *auto*

have $set\text{-}lebesgue\text{-}integral\ M\ A\ f = enn2real\ (set\text{-}nn\text{-}integral\ M\ A\ f)$
unfolding *set-lebesgue-integral-def* **using** *assms*(*1*,*4*) ∗ *eq*
by (*subst integral-eq-nn-integral*)
(*auto intro*!: *nn-integral-cong simp*: *indicator-def of-bool-def set-integrable-def mult-ac*)
also have $\ldots = x$ **using** *assms* **by** *simp*
finally show $set\text{-}lebesgue\text{-}integral\ M\ A\ f = x$ **.**
qed

lemma *set-integral-0* [*simp*, *intro*]: $set\text{-}integrable\ M\ A\ (\lambda y.\ 0)$
by (*simp add*: *set-integrable-def*)

lemma *set-integrable-sum*:
fixes f :: - ⇒ - ⇒ - :: {*banach*, *second-countable-topology*}

assumes *finite B*
assumes $\bigwedge x.\ x \in B \Longrightarrow$ *set-integrable M A* (*f x*)
shows *set-integrable M A* ($\lambda y.\ \sum x{\in}B.\ f\ x\ y$)
using *assms* **by** (*induction rule*: *finite-induct*) (*auto intro*!: *set-integral-add*)

lemma *set-integral-sum*:
fixes *f* :: - ⇒ - ⇒ - :: {*banach*, *second-countable-topology*}
assumes *finite B*
assumes $\bigwedge x.\ x \in B \Longrightarrow$ *set-integrable M A* (*f x*)
shows *set-lebesgue-integral M A* ($\lambda y.\ \sum x{\in}B.\ f\ x\ y$) = ($\sum x{\in}B.$ *set-lebesgue-integral M A* (*f x*))
using *assms*
apply (*induction rule*: *finite-induct*)
apply *simp*
apply *simp*
apply (*subst set-integral-add*)
apply (*auto intro*!: *set-integrable-sum*)
done

lemma *set-nn-integral-cong*:
assumes $M = M'$ $A = B$ $\bigwedge x.\ x \in$ *space M* $\cap A \Longrightarrow f\ x = g\ x$
shows *set-nn-integral M A f* = *set-nn-integral M' B g*
using *assms* **unfolding** *assms*(*1*,*2*) **by** (*intro nn-integral-cong*) (*auto simp*: *indicator-def*)

lemma *set-nn-integral-mono*:
assumes $\bigwedge x.\ x \in$ *space M* $\cap A \Longrightarrow f\ x \le g\ x$
shows *set-nn-integral M A f* ≤ *set-nn-integral M A g*
using *assms* **by** (*intro nn-integral-mono*) (*auto simp*: *indicator-def*)

lemma
fixes *f* :: *real* ⇒ *real*
assumes $a \le b$
assumes *deriv*: $\bigwedge x.\ a \le x \Longrightarrow x \le b \Longrightarrow$ (*F has-field-derivative f x*) (*at x within* {*a*..*b*})
assumes *cont*: *continuous-on* {*a*..*b*} *f*
shows *has-bochner-integral-FTC-Icc-real*:
has-bochner-integral lborel ($\lambda x.\ f\ x *$ *indicator* {*a* .. *b*} *x*) (*F b* − *F a*) (**is** *?has*)
and *integral-FTC-Icc-real*: ($\int x.\ f\ x *$ *indicator* {*a* .. *b*} *x* ∂*lborel*) = *F b* − *F a* (**is** *?eq*)
proof −
have *1*: $\bigwedge x.\ a \le x \Longrightarrow x \le b \Longrightarrow$ (*F has-vector-derivative f x*) (*at x within* {*a* .. *b*})
unfolding *has-real-derivative-iff-has-vector-derivative*[*symmetric*]
using *deriv* **by** *auto*
show *?has ?eq*
using *has-bochner-integral-FTC-Icc*[*OF* ‹*a* ≤ *b*› *1 cont*] *integral-FTC-Icc*[*OF* ‹*a* ≤ *b*› *1 cont*]

by (*auto simp*: *mult.commute*)
qed

lemma *integral-by-parts-integrable*:
fixes *f g F G*::*real* ⇒ *real*
assumes $a \leq b$
assumes *cont-f*[*intro*]: *continuous-on* {*a*..*b*} *f*
assumes *cont-g*[*intro*]: *continuous-on* {*a*..*b*} *g*
assumes [*intro*]: ⋀*x*. *x* ∈ {*a*..*b*} ⟹ (*F has-field-derivative f x*) (*at x within* {*a*..*b*})
assumes [*intro*]: ⋀*x*. *x* ∈ {*a*..*b*} ⟹ (*G has-field-derivative g x*) (*at x within* {*a*..*b*})
shows *integrable lborel* (λ*x*.((*F x*) ∗ (*g x*) + (*f x*) ∗ (*G x*)) ∗ *indicator* {*a* .. *b*} *x*)
proof −
have *integrable lborel* (λ*x*. *indicator* {*a*..*b*} *x* $*_R$ ((*F x*) ∗ (*g x*) + (*f x*) ∗ (*G x*)))
by (*intro borel-integrable-compact continuous-intros assms*)
(*auto intro*!: *DERIV-continuous-on assms*)
thus *?thesis* **by** (*simp add*: *mult-ac*)
qed

lemma *integral-by-parts*:
fixes *f g F G*::*real* ⇒ *real*
assumes [*arith*]: $a \leq b$
assumes *cont-f*[*intro*]: *continuous-on* {*a*..*b*} *f*
assumes *cont-g*[*intro*]: *continuous-on* {*a*..*b*} *g*
assumes [*intro*]: ⋀*x*. *x* ∈ {*a*..*b*} ⟹ (*F has-field-derivative f x*) (*at x within* {*a*..*b*})
assumes [*intro*]: ⋀*x*. *x* ∈ {*a*..*b*} ⟹ (*G has-field-derivative g x*) (*at x within* {*a*..*b*})
shows (∫ *x*. (*F x* ∗ *g x*) ∗ *indicator* {*a* .. *b*} *x* ∂*lborel*)
= *F b* ∗ *G b* − *F a* ∗ *G a* − ∫ *x*. (*f x* ∗ *G x*) ∗ *indicator* {*a* .. *b*} *x* ∂*lborel*
proof−
have *0*: (∫ *x*. (*F x* ∗ *g x* + *f x* ∗ *G x*) ∗ *indicator* {*a* .. *b*} *x* ∂*lborel*) = *F b* ∗ *G b* − *F a* ∗ *G a*
by (*rule integral-FTC-Icc-real*, *auto intro*!: *derivative-eq-intros continuous-intros*)
(*auto intro*!: *assms DERIV-continuous-on*)
have [*continuous-intros*]: *continuous-on* {*a*..*b*} *F*
by (*rule DERIV-continuous-on assms*)+
have [*continuous-intros*]: *continuous-on* {*a*..*b*} *G*
by (*rule DERIV-continuous-on assms*)+

have (∫ *x*. *indicator* {*a*..*b*} *x* $*_R$ (*F x* ∗ *g x* + *f x* ∗ *G x*) ∂*lborel*) =
(∫ *x*. *indicator* {*a*..*b*} *x* $*_R$(*F x* ∗ *g x*) ∂*lborel*) + ∫ *x*. *indicator* {*a*..*b*} *x* $*_R$ (*f x* ∗ *G x*) ∂*lborel*
apply (*subst Bochner-Integration.integral-add*[*symmetric*])
apply (*rule borel-integrable-compact*; *force intro*!: *continuous-intros assms*)
apply (*rule borel-integrable-compact*; *force intro*!: *continuous-intros assms*)
apply (*simp add*: *algebra-simps*)

done

thus *?thesis* **using** *0* **by** (*simp add*: *algebra-simps*)
qed

lemma *interval-lebesgue-integral-by-parts*:
assumes $a \leq b$
assumes *cont-f*[*intro*]: *continuous-on* {*a..b*} *f*
assumes *cont-g*[*intro*]: *continuous-on* {*a..b*} *g*
assumes [*intro*]: $\bigwedge x.\ x \in \{a..b\} \Longrightarrow$ (*F has-field-derivative f x*) (*at x within* {*a..b*})
assumes [*intro*]: $\bigwedge x.\ x \in \{a..b\} \Longrightarrow$ (*G has-field-derivative g x*) (*at x within* {*a..b*})
shows (*LBINT x=a..b. F x* ∗ *g x*) = *F b* ∗ *G b* − *F a* ∗ *G a* − (*LBINT x=a..b. f x* ∗ *G x*)
using *integral-by-parts*[*of a b f g F G*] *assms*
by (*simp add*: *interval-integral-Icc set-lebesgue-integral-def mult-ac*)

lemma *interval-lebesgue-integral-by-parts-01*:
assumes *cont-f*[*intro*]: *continuous-on* {*0..1*} *f*
assumes *cont-g*[*intro*]: *continuous-on* {*0..1*} *g*
assumes [*intro*]: $\bigwedge x.\ x \in \{0..1\} \Longrightarrow$ (*F has-field-derivative f x*) (*at x within* {*0..1*})
assumes [*intro*]: $\bigwedge x.\ x \in \{0..1\} \Longrightarrow$ (*G has-field-derivative g x*) (*at x within* {*0..1*})
shows (*LBINT x=0..1. F x* ∗ *g x*) = *F 1* ∗ *G 1* − *F 0* ∗ *G 0* − (*LBINT x=0..1. f x* ∗ *G x*)
using *interval-lebesgue-integral-by-parts*[*of 0 1 f g F G*] *assms*
by (*simp add*: *zero-ereal-def one-ereal-def*)

lemma *continuous-on-imp-set-integrable-cbox*:
fixes *h* :: *'a* :: *euclidean-space* ⇒ *real*
assumes *continuous-on* (*cbox a b*) *h*
shows *set-integrable lborel* (*cbox a b*) *h*
proof −
from *assms* **have** *h absolutely-integrable-on cbox a b*
by (*rule absolutely-integrable-continuous*)
moreover have (λx. *indicat-real* (*cbox a b*) *x* $*_R$ *h x*) ∈ *borel-measurable borel*
by (*rule borel-measurable-continuous-on-indicator*) (*use assms* **in** *auto*)
ultimately show *?thesis*
unfolding *set-integrable-def* **using** *assms* **by** (*subst* (*asm*) *integrable-completion*) *auto*
qed

1.3 Shifted Legendre polynomials

The first ingredient we need to show Apéry's theorem is the *shifted Legendre polynomials*

$$P_n(X) = \frac{1}{n!}\frac{\partial^n}{\partial X^n}(X^n(1-X)^n)$$

and the auxiliary polynomials

$$Q_{n,k}(X) = \frac{\partial^k}{\partial X^k}(X^n(1-X)^n)\ .$$

Note that P_n is in fact an *integer* polynomial.

Only some very basic properties of these will be proven, since that is all we will need.

```
context
  fixes n :: nat
begin

definition gen-shleg-poly :: nat ⇒ int poly where
  gen-shleg-poly k = (pderiv ^^ k) ([:0, 1, −1:] ^ n)

definition shleg-poly where shleg-poly = gen-shleg-poly n div [:fact n:]
```

We can easily prove the following more explicit formula for $Q_{n,k}$:

$$Q_{n,k}(X) = \sum_{i=0}^{k}(-1)^{k-1}\binom{k}{i} n^{\underline{i}}\, n^{\underline{k-i}}\, X^{n-i}\,(1-X)^{n-k+i}$$

```
lemma gen-shleg-poly-altdef:
  assumes k ≤ n
  shows gen-shleg-poly k =
          (∑ i≤k. smult ((−1)^(k−i) ∗ of-nat (k choose i) ∗
                 pochhammer (n−i+1) i ∗ pochhammer (n−k+i+1) (k−i))
                 ([:0, 1:] ^ (n − i) ∗ [:1, −1:] ^ (n−k+i)))
proof −
  have ∗: (pderiv ^^ i) (x ∘p [:1, −1:]) =
          smult ((−1) ^ i) ((pderiv ^^ i) x ∘p [:1, −1:]) for i and x :: int poly
    by (induction i arbitrary: x)
       (auto simp: pderiv-smult pderiv-pcompose funpow-Suc-right pderiv-pCons
             higher-pderiv-minus simp del: funpow.simps(2))

  have gen-shleg-poly k = (pderiv ^^ k) ([:0, 1, −1:] ^ n)
    by (simp add: gen-shleg-poly-def)
  also have [:0, 1, −1::int:] = [:0, 1:] ∗ [:1, −1:]
    by simp
  also have ... ^ n = [:0, 1:] ^ n ∗ [:1, −1:] ^ n
    by (simp flip: power-mult-distrib)
  also have (pderiv ^^ k) ... =
```

(∑ *i*≤*k*. *smult* (*of-nat* (*k choose i*)) ((*pderiv* ^^ *i*)
([:*0*, *1*:] ^ *n*) ∗ (*pderiv* ^^ (*k* − *i*)) ([:*1*, −*1*:] ^ *n*)))
by (*simp add*: *higher-pderiv-mult*)
also have ... = (∑ *i*≤*k*. *smult* (*of-nat* (*k choose i*))
((*pderiv* ^^ *i*) (*monom 1 n*) ∗ (*pderiv* ^^ (*k* − *i*)) (*monom 1 n* $\circ_p$ [:*1*, −*1*:])))
by (*simp add*: *monom-altdef hom-distribs*)
also have ... = (∑ *i*≤*k*. *smult* ((−*1*) ^ (*k* − *i*) ∗ *of-nat* (*k choose i*))
((*pderiv* ^^ *i*) (*monom 1 n*) ∗ ((*pderiv* ^^ (*k* − *i*)) (*monom 1 n*) $\circ_p$ [:*1*, −*1*:])))
by (*simp add*: ∗ *mult-ac*)
also have ... = (∑ *i*≤*k*. *smult* ((−*1*) ^ (*k* − *i*) ∗ *of-nat* (*k choose i*))
(*monom* (*pochhammer* (*n* − *i* + *1*) *i*) (*n* − *i*) ∗
monom (*pochhammer* (*n* − *k* + *i* + *1*) (*k* − *i*)) (*n* − *k* + *i*) $\circ_p$ [:*1*, −*1*:]))
using *assms* **by** (*simp add*: *higher-pderiv-monom*)
also have ... = (∑ *i*≤*k*. *smult* ((−*1*) ^ (*k* − *i*) ∗ *of-nat* (*k choose i*) ∗ *pochhammer* (*n* − *i* + *1*) *i* ∗ *pochhammer* (*n* − *k* + *i* + *1*) (*k* − *i*))
([:*0*, *1*:] ^ (*n* − *i*) ∗ [:*1*, −*1*:] ^ (*n* − *k* + *i*)))
by (*simp add*: *monom-altdef algebra-simps pcompose-smult hom-distribs*)
finally show *?thesis* **.**
qed

lemma *degree-gen-shleg-poly* [*simp*]: *degree* (*gen-shleg-poly k*) = *2* ∗ *n* − *k*
by (*simp add*: *gen-shleg-poly-def degree-higher-pderiv degree-power-eq*)

lemma *gen-shleg-poly-n*: *gen-shleg-poly n* = *smult* (*fact n*) *shleg-poly*
proof −
obtain *r* **where** *r*: *gen-shleg-poly n* = [:*fact n*:] ∗ *r*
unfolding *gen-shleg-poly-def* **using** *fact-dvd-higher-pderiv*[*of n* [:*0*,*1*,−*1*:]^*n*] **by** *blast*
have *smult* (*fact n*) *shleg-poly* = *smult* (*fact n*) (*gen-shleg-poly n div* [:*fact n*:])
by (*simp add*: *shleg-poly-def*)
also note *r*
also have [:*fact n*:] ∗ *r div* [:*fact n*:] = *r*
by (*rule nonzero-mult-div-cancel-left*) *auto*
finally show *?thesis*
by (*simp add*: *r*)
qed

lemma *degree-shleg-poly* [*simp*]: *degree shleg-poly* = *n*
using *degree-gen-shleg-poly*[*of n*] **by** (*simp add*: *gen-shleg-poly-n*)

lemma *pderiv-gen-shleg-poly* [*simp*]: *pderiv* (*gen-shleg-poly k*) = *gen-shleg-poly* (*Suc k*)
by (*simp add*: *gen-shleg-poly-def*)

The following functions are the interpretation of the shifted Legendre polynomials and the auxiliary polynomials as a function from reals to reals.

definition *Gen-Shleg* :: *nat* ⇒ *real* ⇒ *real*
  where *Gen-Shleg k x* = *poly* (*of-int-poly* (*gen-shleg-poly k*)) *x*

definition *Shleg* :: *real* ⇒ *real* **where** *Shleg* = *poly* (*of-int-poly shleg-poly*)

lemma *Gen-Shleg-altdef*:
  assumes $k \leq n$
  shows *Gen-Shleg k x* = ($\sum$ *i*≤*k*. (−*1*)^(*k*−*i*) ∗ *of-nat* (*k choose i*) ∗
    of-int (*pochhammer* (*n*−*i*+*1*) *i* ∗ *pochhammer* (*n*−*k*+*i*+*1*) (*k*−*i*)) ∗
    x ^ (*n* − *i*) ∗ (*1* − *x*) ^ (*n*−*k*+*i*))
  using *assms* **by** (*simp add*: *Gen-Shleg-def gen-shleg-poly-altdef poly-sum mult-ac hom-distribs*)

lemma *Gen-Shleg-0* [*simp*]: *k* < *n* ⟹ *Gen-Shleg k 0* = *0*
  by (*simp add*: *Gen-Shleg-altdef zero-power*)

lemma *Gen-Shleg-1* [*simp*]: *k* < *n* ⟹ *Gen-Shleg k 1* = *0*
  by (*simp add*: *Gen-Shleg-altdef zero-power*)

lemma *Gen-Shleg-n-0* [*simp*]: *Gen-Shleg n 0* = *fact n*
proof −
  have *Gen-Shleg n 0* = ($\sum$ *i*≤*n*. (−*1*) ^ (*n*−*i*) ∗ *real* (*n choose i*) ∗
    (*real* (*pochhammer* (*Suc* (*n*−*i*)) *i*) ∗
    real (*pochhammer* (*Suc i*) (*n*−*i*))) ∗ *0* ^ (*n* − *i*))
    by (*simp add*: *Gen-Shleg-altdef*)
  also have . . . = ($\sum$ *i*∈{*n*}. (−*1*) ^ (*n*−*i*) ∗ *real* (*n choose i*) ∗
    (*real* (*pochhammer* (*Suc* (*n*−*i*)) *i*) ∗
    real (*pochhammer* (*Suc i*) (*n*−*i*))) ∗ *0* ^ (*n* − *i*))
    by (*intro sum.mono-neutral-right*) *auto*
  also have . . . = *fact n*
    by (*simp add*: *pochhammer-fact flip*: *pochhammer-of-nat*)
  finally show *?thesis* **.**
qed

lemma *Gen-Shleg-n-1* [*simp*]: *Gen-Shleg n 1* = (−*1*) ^ *n* ∗ *fact n*
proof −
  have *Gen-Shleg n 1* = ($\sum$ *i*≤*n*. (−*1*) ^ (*n*−*i*) ∗ *real* (*n choose i*) ∗
    (*real* (*pochhammer* (*Suc* (*n*−*i*)) *i*) ∗
    real (*pochhammer* (*Suc i*) (*n*−*i*))) ∗ *0* ^ *i*)
    by (*simp add*: *Gen-Shleg-altdef*)
  also have . . . = ($\sum$ *i*∈{*0*}. (−*1*) ^ (*n*−*i*) ∗ *real* (*n choose i*) ∗
    (*real* (*pochhammer* (*Suc* (*n*−*i*)) *i*) ∗
    real (*pochhammer* (*Suc i*) (*n*−*i*))) ∗ *0* ^ *i*)
    by (*intro sum.mono-neutral-right*) *auto*
  also have . . . = (−*1*) ^ *n* ∗ *fact n*
    by (*simp add*: *pochhammer-fact flip*: *pochhammer-of-nat*)
  finally show *?thesis* **.**
qed

```
lemma Shleg-altdef: Shleg x = Gen-Shleg n x / fact n
  by (simp add: Shleg-def Gen-Shleg-def gen-shleg-poly-n hom-distribs)

lemma Shleg-0 [simp]: Shleg 0 = 1 and Shleg-1 [simp]: Shleg 1 = (−1) ^ n
  by (simp-all add: Shleg-altdef)

lemma Gen-Shleg-0-left: Gen-Shleg 0 x = x ^ n ∗ (1 − x) ^ n
  by (simp add: Gen-Shleg-def gen-shleg-poly-def power-mult-distrib hom-distribs)

lemma has-field-derivative-Gen-Shleg:
  (Gen-Shleg k has-field-derivative Gen-Shleg (Suc k) x) (at x)
proof −
  note [derivative-intros] = poly-DERIV
  show ?thesis unfolding Gen-Shleg-def
   by (rule derivative-eq-intros refl)+ (auto simp: hom-distribs simp flip: of-int-hom.map-poly-pderiv)
qed

lemma continuous-on-Gen-Shleg: continuous-on A (Gen-Shleg k)
  by (auto simp: Gen-Shleg-def intro!: continuous-intros)

lemma continuous-on-Gen-Shleg' [continuous-intros]:
  continuous-on A f ⟹ continuous-on A (λx. Gen-Shleg k (f x))
  by (rule continuous-on-compose2[OF continuous-on-Gen-Shleg[of UNIV]]) auto

lemma continuous-on-Shleg: continuous-on A Shleg
  by (auto simp: Shleg-def intro!: continuous-intros)

lemma continuous-on-Shleg' [continuous-intros]:
  continuous-on A f ⟹ continuous-on A (λx. Shleg (f x))
  by (rule continuous-on-compose2[OF continuous-on-Shleg[of UNIV]]) auto

lemma measurable-Gen-Shleg [measurable]: Gen-Shleg n ∈ borel-measurable borel
  by (intro borel-measurable-continuous-onI continuous-on-Gen-Shleg)

lemma measurable-Shleg [measurable]: Shleg ∈ borel-measurable borel
  by (intro borel-measurable-continuous-onI continuous-on-Shleg)

end
```

1.4 Auxiliary facts about the ζ function

```
lemma Re-zeta-ge-1:
  assumes x > 1
  shows   Re (zeta (of-real x)) ≥ 1
proof −
  have ∗: (λn. real (Suc n) powr −x) sums Re (zeta (complex-of-real x))
    using sums-Re[OF sums-zeta[of of-real x]] assms by (simp add: powr-Reals-eq)
  show Re (zeta (of-real x)) ≥ 1
```

proof (*rule sums-le*[*OF* - - ∗])
show (λ*n*. *if n* = *0 then 1 else 0*) *sums 1*
by (*rule sums-single*)
qed *auto*
qed

lemma *sums-zeta-of-nat-offset*:
fixes *r* :: *nat*
assumes *n*: *n* > *1*
shows (λ*k*. *1* / (*r* + *k* + *1*) ^ *n*) *sums* (*zeta* (*of-nat n*) − (∑*k*=*1*..*r*. *1* / *k* ^ *n*))
proof −
have (λ*k*. *1* / (*k* + *1*) ^ *n*) *sums zeta* (*of-nat n*)
using *sums-zeta*[*of of-nat n*] *n*
by (*simp add*: *powr-minus field-simps flip*: *of-nat-Suc*)
from *sums-split-initial-segment*[*OF this*, *of r*]
have (λ*k*. *1* / (*r* + *k* + *1*) ^ *n*) *sums* (*zeta* (*of-nat n*) − (∑*k*<*r*. *1* / *Suc k* ^ *n*))
by (*simp add*: *algebra-simps*)
also have (∑*k*<*r*. *1* / *Suc k* ^ *n*) = (∑*k*=*1*..*r*. *1* / *k* ^ *n*)
by (*intro sum.reindex-bij-witness*[*of* - λ*k*. *k* − *1 Suc*]) *auto*
finally show *?thesis* **.**
qed

lemma *sums-Re-zeta-of-nat-offset*:
fixes *r* :: *nat*
assumes *n*: *n* > *1*
shows (λ*k*. *1* / (*r* + *k* + *1*) ^ *n*) *sums* (*Re* (*zeta* (*of-nat n*)) − (∑*k*=*1*..*r*. *1* / *k* ^ *n*))
proof −
have (λ*k*. *Re* (*1* / (*r* + *k* + *1*) ^ *n*)) *sums* (*Re* (*zeta* (*of-nat n*) − (∑*k*=*1*..*r*. *1* / *k* ^ *n*)))
by (*intro sums-Re sums-zeta-of-nat-offset assms*)
thus *?thesis* **by** *simp*
qed

1.5 Divisor of a sum of rationals

A finite sum of rationals of the form $\frac{a_1}{b_1} + \ldots + \frac{a_n}{b_n}$ can be brought into the form $\frac{c}{d}$, where d is the LCM of the b_i (or some integer multiple thereof).

lemma *sum-rationals-common-divisor*:
fixes *f g* :: *'a* ⇒ *int*
assumes *finite A*
assumes ⋀*x*. *x* ∈ *A* ⟹ *g x* ≠ *0*
shows ∃ *c*. (∑*x*∈*A*. *f x* / *g x*) = *real-of-int c* / (*LCM x*∈*A*. *g x*)
using *assms*
proof (*induction rule*: *finite-induct*)
case *empty*
thus *?case* **by** *auto*

```
next
  case (insert x A)
  define d where d = (LCM x∈A. g x)
  from insert have [simp]: d ≠ 0
    by (auto simp: d-def Lcm-0-iff)
  from insert have [simp]: g x ≠ 0 by auto
  from insert obtain c where c: (∑ x∈A. f x / g x) = real-of-int c / real-of-int d
    by (auto simp: d-def)
  define e1 where e1 = lcm d (g x) div d
  define e2 where e2 = lcm d (g x) div g x
  have (∑ y∈insert x A. f y / g y) = c / d + f x / g x
    using insert c by simp
  also have c / d = (c ∗ e1) / lcm d (g x)
    by (simp add: e1-def real-of-int-div)
  also have f x / g x = (f x ∗ e2) / lcm d (g x)
    by (simp add: e2-def real-of-int-div)
  also have (c ∗ e1) / lcm d (g x) + ... = (c ∗ e1 + f x ∗ e2) / (LCM x∈insert
x A. g x)
    using insert by (simp add: add-divide-distrib lcm.commute d-def)
  finally show ?case ..
qed

lemma sum-rationals-common-divisor':
  fixes f g :: 'a ⇒ int
  assumes finite A
  assumes ⋀x. x ∈ A ⟹ g x ≠ 0 and (⋀x. x ∈ A ⟹ g x dvd d) and d ≠ 0
  shows   ∃ c. (∑ x∈A. f x / g x) = real-of-int c / real-of-int d
proof −
  define d' where d' = (LCM x∈A. g x)
  have d' dvd d
    unfolding d'-def using assms(3) by (auto simp: Lcm-dvd-iff)
  then obtain e where e: d = d' ∗ e by blast
  have ∃ c. (∑ x∈A. f x / g x) = real-of-int c / (LCM x∈A. g x)
    by (rule sum-rationals-common-divisor) fact+
  then obtain c where c: (∑ x∈A. f x / g x) = real-of-int c / real-of-int d'
    unfolding d'-def ..
  also have ... = real-of-int (c ∗ e) / real-of-int d
    using ‹d ≠ 0› by (simp add: e)
  finally show ?thesis ..
qed
```

1.6 The first double integral

We shall now investigate the double integral

$$I_1 := \int_0^1 \int_0^1 -\frac{\ln(xy)}{1-xy}\, x^r y^s \,\mathrm{d}x\,\mathrm{d}y \ .$$

Since everything is non-negative for now, we can work over the extended non-negative real numbers and the issues of integrability or summability do not arise at all.

definition *beukers-nn-integral1* :: *nat* ⇒ *nat* ⇒ *ennreal* **where**
  beukers-nn-integral1 r s =
    (∫⁺(*x*,*y*)∈{*0*<..<*1*}×{*0*<..<*1*}. *ennreal* (−*ln* (*x*∗*y*) / (*1* − *x*∗*y*) ∗ *x*^*r* ∗ *y*^*s*) ∂*lborel*)

definition *beukers-integral1* :: *nat* ⇒ *nat* ⇒ *real* **where**
  beukers-integral1 r s = (∫ (*x*,*y*)∈{*0*<..<*1*}×{*0*<..<*1*}. (−*ln* (*x*∗*y*) / (*1* − *x*∗*y*) ∗ *x*^*r* ∗ *y*^*s*) ∂*lborel*)

lemma
  fixes *x y z* :: *real*
  assumes *xyz*: *x* ∈ {*0*<..<*1*} *y* ∈ {*0*<..<*1*} *z* ∈ {*0*..*1*}
  shows *beukers-denom-ineq*: (*1* − *x* ∗ *y*) ∗ *z* < *1* **and** *beukers-denom-neq*: (*1* − *x* ∗ *y*) ∗ *z* ≠ *1*
proof −
  from *xyz* **have** ∗: *x* ∗ *y* < *1* ∗ *1*
    by (*intro mult-strict-mono*) *auto*
  from ∗ **have** (*1* − *x* ∗ *y*) ∗ *z* ≤ (*1* − *x* ∗ *y*) ∗ *1*
    using *xyz* **by** (*intro mult-left-mono*) *auto*
  also have . . . < *1* ∗ *1*
    using *xyz* **by** (*intro mult-strict-right-mono*) *auto*
  finally show (*1* − *x* ∗ *y*) ∗ *z* < *1* (*1* − *x* ∗ *y*) ∗ *z* ≠ *1* **by** *simp-all*
qed

We first evaluate the improper integral

$$\int_0^1 -\ln x \cdot x^e \,\mathrm{d}x = \frac{1}{(e+1)^2} \ .$$

for any $e > -1$.

lemma *integral-0-1-ln-times-powr*:
  assumes *e* > −*1*
  shows (*LBINT x*=*0*..*1*. −*ln x* ∗ *x powr e*) = *1* / (*e* + *1*)²
    and *interval-lebesgue-integrable lborel 0 1* (λ*x*. −*ln x* ∗ *x powr e*)
proof −
  define *f* **where** *f* = (λ*x*. −*ln x* ∗ *x powr e*)
  define *F* **where** *F* = (λ*x*. *x powr* (*e* + *1*) ∗ (*1* − (*e* + *1*) ∗ *ln x*) / (*e* + *1*) ^ *2*)

  have *0*: *isCont f x* **if** *x* ∈ {*0*<..<*1*} **for** *x*
    using *that* **by** (*auto intro*!: *continuous-intros simp*: *f-def*)
  have *1*: (*F has-real-derivative f x*) (*at x*) **if** *x* ∈ {*0*<..<*1*} **for** *x*
  proof −
    show (*F has-real-derivative f x*) (*at x*)
      unfolding *F-def f-def* **using** *that assms*
      apply (*insert that assms*)
      apply (*rule derivative-eq-intros refl* | *simp*)+
      apply (*simp add*: *divide-simps*)

```
    apply (simp add: power2-eq-square algebra-simps powr-add power-numeral-reduce)
     done
  qed
  have 2: AE x in lborel. ereal 0 < ereal x ⟶ ereal x < ereal 1 ⟶ 0 ≤ f x
    by (intro AE-I2) (auto simp: f-def mult-nonpos-nonneg)
  have 3: ((F ∘ real-of-ereal) ⟶ 0) (at-right (ereal 0))
    unfolding ereal-tendsto-simps F-def using assms by real-asymp
  have 4: ((F ∘ real-of-ereal) ⟶ F 1) (at-left (ereal 1))
    unfolding ereal-tendsto-simps F-def
    using assms by real-asymp (simp add: field-simps)

  have (LBINT x=ereal 0..ereal 1. f x) = F 1 − 0
    by (rule interval-integral-FTC-nonneg[where F = F])
       (use 0 1 2 3 4 in auto)
  thus (LBINT x=0..1. −ln x ∗ x powr e) = 1 / (e + 1)^2
    by (simp add: F-def zero-ereal-def one-ereal-def f-def)
  have set-integrable lborel (einterval (ereal 0) (ereal 1)) f
    by (rule interval-integral-FTC-nonneg)
       (use 0 1 2 3 4 in auto)
  thus interval-lebesgue-integrable lborel 0 1 f
    by (simp add: interval-lebesgue-integrable-def einterval-def)
qed

lemma interval-lebesgue-integral-lborel-01-cong:
  assumes ⋀x. x ∈ {0<..<1} ⟹ f x = g x
  shows   interval-lebesgue-integral lborel 0 1 f =
          interval-lebesgue-integral lborel 0 1 g
  using assms
  by (subst (1 2) interval-integral-Ioo)
     (auto intro!: set-lebesgue-integral-cong assms)

lemma nn-integral-0-1-ln-times-powr:
  assumes e > −1
  shows   (∫⁺y∈{0<..<1}. ennreal (−ln y ∗ y powr e) ∂lborel) = ennreal (1 /
(e + 1)^2)
proof −
  have ∗: (LBINT x=0..1. − ln x ∗ x powr e = 1 / (e + 1)^2)
          interval-lebesgue-integrable lborel 0 1 (λx. − ln x ∗ x powr e)
    using integral-0-1-ln-times-powr[OF assms] by simp-all
  have eq: (λy. (if 0 < y ∧ y < 1 then 1 else 0) ∗ ln y ∗ y powr e) =
             (λy. if 0 < y ∧ y < 1 then ln y ∗ y powr e else 0)
    by auto

  have (∫⁺y∈{0<..<1}. ennreal (−ln y ∗ y powr e) ∂lborel) =
          (∫⁺y. ennreal (−ln y ∗ y powr e ∗ indicator {0<..<1} y) ∂lborel)
    by (intro nn-integral-cong) (auto simp: indicator-def)
  also have ... = ennreal (1 / (e + 1)^2)
    using ∗ eq
    by (subst nn-integral-eq-integral)
```

(*auto intro*!: *AE-I2 simp*: *indicator-def interval-lebesgue-integrable-def set-integrable-def one-ereal-def zero-ereal-def interval-integral-Ioo mult-ac mult-nonpos-nonneg set-lebesgue-integral-def*)
finally show *?thesis* **.**
qed

lemma *nn-integral-0-1-ln-times-power*:
$(\int^{+} y \in \{0<..<1\}.\ ennreal\ (-ln\ y * y\ \widehat{}\ n)\ \partial lborel) = ennreal\ (1\ /\ (n + 1)^2)$
proof −
have $(\int^{+} y \in \{0<..<1\}.\ ennreal\ (-ln\ y * y\ \widehat{}\ n)\ \partial lborel) =$
$(\int^{+} y \in \{0<..<1\}.\ ennreal\ (-ln\ y * y\ powr\ real\ n)\ \partial lborel)$
by (*intro set-nn-integral-cong*) (*auto simp*: *powr-realpow*)
also have ... = *ennreal* (*1* / (*n* + *1*)^*2*)
by (*subst nn-integral-0-1-ln-times-powr*) *auto*
finally show *?thesis* **by** *simp*
qed

Next, we also evaluate the more trivial integral

$$\int_0^1 x^n\ \mathrm{d}x\ .$$

lemma *nn-integral-0-1-power*:
$(\int^{+} y \in \{0<..<1\}.\ ennreal\ (y\ \widehat{}\ n)\ \partial lborel) = ennreal\ (1\ /\ (n + 1))$
proof −
have ∗: ((*λa*. *a* ^(*n* + *1*) / *real* (*n* + *1*)) *has-real-derivative x* ^ *n*) (*at x*) **for** *x*
by (*rule derivative-eq-intros refl* | *simp*)+
have $(\int^{+} y \in \{0<..<1\}.\ ennreal\ (y\ \widehat{}\ n)\ \partial lborel) = (\int^{+} y \in \{0..1\}.\ ennreal\ (y\ \widehat{}\ n)\ \partial lborel)$
by (*intro nn-integral-cong-AE AE-I*[*of* - - {*0*,*1*}])
(*auto simp*: *indicator-def emeasure-lborel-countable*)
also have ... = *ennreal* (*1* ^(*n* + *1*) / (*n* + *1*) − *0* ^(*n* + *1*) / (*n* + *1*))
using ∗ **by** (*intro nn-integral-FTC-Icc*) *auto*
also have ... = *ennreal* (*1* / (*n* + *1*))
by *simp*
finally show *?thesis* **by** *simp*
qed

I_1 can alternatively be written as the triple integral

$$\int_0^1 \int_0^1 \int_0^1 \frac{x^r y^s}{1-(1-xy)w}\ \mathrm{d}x\, \mathrm{d}y\, \mathrm{d}w\ .$$

lemma *beukers-nn-integral1-altdef*:
beukers-nn-integral1 r s =
$(\int^{+} (w,x,y) \in \{0<..<1\} \times \{0<..<1\} \times \{0<..<1\}.$
ennreal (*1* / (*1*−(*1*−*x*∗*y*)∗*w*) ∗ *x*^*r* ∗ *y*^*s*) ∂*lborel*)
proof −
have $(\int^{+} (w,x,y) \in \{0<..<1\} \times \{0<..<1\} \times \{0<..<1\}.$

```
        ennreal (1 / (1−(1−x∗y)∗w) ∗ x^r ∗ y^s) ∂lborel) =
      (∫+(x,y)∈{0<..<1}×{0<..<1}. (∫+w∈{0<..<1}.
        ennreal (1 / (1−(1−x∗y)∗w) ∗ x^r ∗ y^s) ∂lborel) ∂lborel)
    by (subst lborel-prod [symmetric], subst lborel-pair.nn-integral-snd [symmetric])
      (auto simp: case-prod-unfold indicator-def simp flip: lborel-prod intro!: nn-integral-cong)
  also have ... = (∫+(x,y)∈{0<..<1}×{0<..<1}. ennreal (−ln (x∗y)/(1−x∗y)
∗ x^r ∗ y^s) ∂lborel)
  proof (intro nn-integral-cong, clarify)
    fix x y :: real
    have (∫+w∈{0<..<1}. ennreal (1/(1−(1−x∗y)∗w)∗x^r∗y^s) ∂lborel) =
            ennreal (−ln (x∗y)∗x^r∗y^s/(1−x∗y))
      if xy: (x, y) ∈ {0<..<1} × {0<..<1}
    proof −
      from xy have x ∗ y < 1
        using mult-strict-mono[of x 1 y 1] by simp
      have deriv: ((λw. −ln (1−(1−x∗y)∗w) / (1−x∗y)) has-real-derivative
            1/(1−(1−x∗y)∗w)) (at w) if w: w ∈ {0..1} for w
        by (insert xy w ‹x∗y<1› beukers-denom-ineq[of x y w])
          (rule derivative-eq-intros refl | simp add: divide-simps)+
      have (∫+w∈{0<..<1}. ennreal (1/(1−(1−x∗y)∗w)∗x^r∗y^s) ∂lborel) =
              ennreal (x^r∗y^s) ∗ (∫+w∈{0<..<1}. ennreal (1/(1−(1−x∗y)∗w))
∂lborel)
        using xy by (subst nn-integral-cmult [symmetric])
                      (auto intro!: nn-integral-cong simp: indicator-def simp flip:
ennreal-mult′)
      also have (∫+w∈{0<..<1}. ennreal (1/(1−(1−x∗y)∗w)) ∂lborel) =
                (∫+w∈{0..1}. ennreal (1/(1−(1−x∗y)∗w)) ∂lborel)
        by (intro nn-integral-cong-AE AE-I[of - - {0,1}])
          (auto simp: emeasure-lborel-countable indicator-def)
      also have (∫+w∈{0..1}. ennreal (1/(1−(1−x∗y)∗w)) ∂lborel) =
             ennreal (−ln (1−(1−x∗y)∗1)/(1−x∗y) − (−ln (1−(1−x∗y)∗0)/(1−x∗y)))
        using xy deriv less-imp-le[OF beukers-denom-ineq[of x y]]
        by (intro nn-integral-FTC-Icc) auto
      finally show ?thesis using xy
        by (simp flip: ennreal-mult′ ennreal-mult′′ add: mult-ac)
    qed
    thus (∫+w∈{0<..<1}. ennreal (1/(1−(1−x∗y)∗w)∗x^r∗y^s) ∂lborel) ∗ indi-
cator ({0<..<1}×{0<..<1}) (x, y) =
          ennreal (−ln (x∗y)/(1−x∗y)∗x^r∗y^s) ∗ indicator ({0<..<1}×{0<..<1})
(x, y)
      by (auto simp: indicator-def)
  qed
  also have ... = beukers-nn-integral1 r s
    by (simp add: beukers-nn-integral1-def)
  finally show ?thesis ..
qed

context
  fixes r s :: nat and I1 I2′ :: real and I2 :: ennreal and D :: (real × real × real)
```

set
assumes *rs*: *s* ≤ *r*
defines *D* ≡ {*0*<..<*1*}×{*0*<..<*1*}×{*0*<..<*1*}
begin

By unfolding the geometric series, pulling the summation out and evaluating the integrals, we find that

$$I_1 = \sum_{k=0}^{\infty} \frac{1}{(k+r+1)^2(k+s+1)} + \frac{1}{(k+r+1)(k+s+1)^2} \; .$$

lemma *beukers-nn-integral1-series*:
beukers-nn-integral1 r s = (∑ *k*. *ennreal* (*1*/((*k*+*r*+*1*)^*2*∗(*k*+*s*+*1*)) + *1*/((*k*+*r*+*1*)∗(*k*+*s*+*1*)^*2*)))
proof −
have *beukers-nn-integral1 r s* =
(∫ ⁺(*x*,*y*)∈{*0*<..<*1*}×{*0*<..<*1*}. (∑ *k*. *ennreal* (−*ln* (*x*∗*y*) ∗ *x*^(*k*+*r*) ∗ *y*^(*k*+*s*))) ∂*lborel*)
unfolding *beukers-nn-integral1-def*
proof (*intro set-nn-integral-cong refl*, *clarify*)
fix *x y* :: *real* **assume** *xy*: *x* ∈ {*0*<..<*1*} *y* ∈ {*0*<..<*1*}
from *xy* **have** *x* ∗ *y* < *1* **using** *mult-strict-mono*[*of x 1 y 1*] **by** *simp*
have (∑ *k*. *ennreal* (−*ln* (*x*∗*y*) ∗ *x*^(*k*+*r*) ∗ *y*^(*k*+*s*))) =
ennreal (−*ln* (*x*∗*y*) ∗ *x*^*r* ∗ *y*^*s*) ∗ (∑ *k*. *ennreal* ((*x*∗*y*)^*k*))
using *xy* **by** (*subst ennreal-suminf-cmult* [*symmetric*], *subst ennreal-mult*″ [*symmetric*])
(*auto simp*: *power-add mult-ac power-mult-distrib*)
also have (∑ *k*. *ennreal* ((*x*∗*y*)^*k*)) = *ennreal* (*1* / (*1* − *x*∗*y*))
using *geometric-sums*[*of x*∗*y*] ‹*x* ∗ *y* < *1*› *xy* **by** (*intro suminf-ennreal-eq*) *auto*
also have *ennreal* (−*ln* (*x*∗*y*) ∗ *x*^*r* ∗ *y*^*s*) ∗ . . . =
ennreal (−*ln* (*x*∗*y*) / (*1* − *x*∗*y*) ∗ *x*^*r* ∗ *y*^*s*)
using ‹*x* ∗ *y* < *1*› **by** (*subst ennreal-mult*″ [*symmetric*]) *auto*
finally show *ennreal* (−*ln* (*x*∗*y*) / (*1* − *x*∗*y*) ∗ *x*^*r* ∗ *y*^*s*) =
(∑ *k*. *ennreal* (−*ln* (*x*∗*y*) ∗ *x*^(*k*+*r*) ∗ *y*^(*k*+*s*))) **..**
qed
also have . . . = (∑ *k*. (∫ ⁺(*x*,*y*)∈{*0*<..<*1*}×{*0*<..<*1*}. (*ennreal* (−*ln* (*x*∗*y*) ∗ *x*^(*k*+*r*) ∗ *y*^(*k*+*s*))) ∂*lborel*))
unfolding *case-prod-unfold* **by** (*subst nn-integral-suminf* [*symmetric*]) (*auto simp flip*: *borel-prod*)
also have . . . = (∑ *k*. *ennreal* (*1* / ((*k*+*r*+*1*)^*2*∗(*k*+*s*+*1*)) + *1* / ((*k*+*r*+*1*)∗(*k*+*s*+*1*)^*2*)))
proof (*rule suminf-cong*)
fix *k* :: *nat*
define *F* **where** *F* = (λ*x y*::*real*. *x* + *y*)
have (∫ ⁺(*x*,*y*)∈{*0*<..<*1*}×{*0*<..<*1*}. *ennreal* (−*ln* (*x*∗*y*) ∗ *x*^(*k*+*r*) ∗ *y*^(*k*+*s*)) ∂*lborel*) =
(∫ ⁺*x*∈{*0*<..<*1*}. (∫ ⁺*y*∈{*0*<..<*1*}. *ennreal* (−*ln* (*x*∗*y*) ∗ *x*^(*k*+*r*) ∗ *y*^(*k*+*s*)) ∂*lborel*) ∂*lborel*)
unfolding *case-prod-unfold lborel-prod* [*symmetric*]
by (*subst lborel.nn-integral-fst* [*symmetric*]) (*auto intro*!: *nn-integral-cong simp*: *indicator-def*)

also have . . . = (∫+x∈{0<..<1}. ennreal (−ln x ∗ x^(k+r) / (k+s+1) +
x^(k+r)/(k+s+1)^2) ∂lborel)
proof (intro set-nn-integral-cong refl, clarify)
fix x :: real **assume** x: x ∈ {0<..<1}
have (∫+y∈{0<..<1}. ennreal (−ln (x∗y) ∗ x^(k+r) ∗ y^(k+s)) ∂lborel) =
(∫+y∈{0<..<1}. (ennreal (−ln x ∗ x^(k+r) ∗ y^(k+s)) + ennreal
(−ln y ∗ x^(k+r) ∗ y^(k+s))) ∂lborel)
by (intro set-nn-integral-cong)
(use x **in** ‹auto simp: ln-mult ring-distribs mult-nonpos-nonneg simp flip:
ennreal-plus›)
also have . . . = (∫+y∈{0<..<1}. ennreal (−ln x ∗ x^(k+r) ∗ y^(k+s))
∂lborel) +
(∫+y∈{0<..<1}. ennreal (−ln y ∗ x^(k+r) ∗ y^(k+s)) ∂lborel)
by (subst nn-integral-add [symmetric]) (auto intro!: nn-integral-cong simp:
indicator-def)
also have (∫+y∈{0<..<1}. ennreal (−ln x ∗ x^(k+r) ∗ y^(k+s)) ∂lborel) =
ennreal (−ln x ∗ x^(k+r)) ∗ (∫+y∈{0<..<1}. ennreal (y^(k+s))
∂lborel)
by (subst nn-integral-cmult [symmetric])
(auto intro!: nn-integral-cong simp: indicator-def simp flip: ennreal-mult'')
also have (∫+y∈{0<..<1}. ennreal (y^(k+s)) ∂lborel) = ennreal (1/(k+s+1))
by (subst nn-integral-0-1-power) simp
also have ennreal (−ln x ∗ x^(k+r)) ∗ . . . = ennreal (−ln x ∗ x^(k+r) /
(k+s+1))
by (subst ennreal-mult'' [symmetric]) auto
also have (∫+y∈{0<..<1}. ennreal (−ln y ∗ x^(k+r) ∗ y^(k+s)) ∂lborel) =
ennreal (x^(k+r)) ∗ (∫+y∈{0<..<1}. ennreal (−ln y ∗ y^(k+s))
∂lborel)
by (subst nn-integral-cmult [symmetric])
(use x **in** ‹auto intro!: nn-integral-cong simp: indicator-def mult-ac simp
flip: ennreal-mult'›)
also have (∫+y∈{0<..<1}. ennreal (−ln y ∗ y^(k+s)) ∂lborel) = ennreal (1
/ (k + s + 1)^2)
by (subst nn-integral-0-1-ln-times-power) simp
also have ennreal (x ^ (k + r)) ∗ . . . = ennreal (x ^ (k + r) / (k + s + 1)
^ 2)
by (subst ennreal-mult'' [symmetric]) auto
also have ennreal (− ln x ∗ x ^ (k + r) / (k + s + 1)) + . . . =
ennreal (−ln x ∗ x^(k+r) / (k+s+1) + x^(k+r)/(k+s+1)^2)
using x **by** (subst ennreal-plus) (auto simp: mult-nonpos-nonneg divide-nonpos-nonneg)
finally show (∫+y∈{0<..<1}. ennreal (−ln (x∗y) ∗ x^(k+r) ∗ y^(k+s))
∂lborel) =
ennreal (−ln x ∗ x^(k+r) / (k+s+1) + x^(k+r)/(k+s+1)^2) **.**
qed
also have . . . = (∫+x∈{0<..<1}. (ennreal (−ln x ∗ x^(k+r) / (k+s+1)) +
ennreal (x^(k+r)/(k+s+1)^2)) ∂lborel)
by (intro set-nn-integral-cong refl, subst ennreal-plus)
(auto simp: mult-nonpos-nonneg divide-nonpos-nonneg)
also have . . . = (∫+x∈{0<..<1}. ennreal (−ln x ∗ x^(k+r) / (k+s+1))

∂lborel) +
(∫ +x∈{0<..<1}. ennreal (x^(k+r)/(k+s+1)^2) ∂lborel)
by (subst nn-integral-add [symmetric]) (auto intro!: nn-integral-cong simp: indicator-def)
also have (∫ +x∈{0<..<1}. ennreal (−ln x ∗ x^(k+r) / (k+s+1)) ∂lborel) =
ennreal (1 / (k+s+1)) ∗ (∫ +x∈{0<..<1}. ennreal (−ln x ∗ x^(k+r)) ∂lborel)
by (subst nn-integral-cmult [symmetric])
(auto intro!: nn-integral-cong simp: indicator-def simp flip: ennreal-mult′)
also have . . . = ennreal (1 / ((k+s+1) ∗ (k+r+1)^2))
by (subst nn-integral-0-1-ln-times-power, subst ennreal-mult [symmetric]) (auto simp: algebra-simps)
also have (∫ +x∈{0<..<1}. ennreal (x^(k+r)/(k+s+1)^2) ∂lborel) =
ennreal (1/(k+s+1)^2) ∗ (∫ +x∈{0<..<1}. ennreal (x^(k+r)) ∂lborel)
by (subst nn-integral-cmult [symmetric])
(auto intro!: nn-integral-cong simp: indicator-def simp flip: ennreal-mult′)
also have . . . = ennreal (1/((k+r+1)∗(k+s+1)^2))
by (subst nn-integral-0-1-power, subst ennreal-mult [symmetric]) (auto simp: algebra-simps)
also have ennreal (1 / ((k+s+1) ∗ (k+r+1)^2)) + . . . =
ennreal (1/((k+r+1)^2∗(k+s+1)) + 1/((k+r+1)∗(k+s+1)^2))
by (subst ennreal-plus [symmetric]) (auto simp: algebra-simps)
finally show (∫ +(x,y)∈{0<..<1}×{0<..<1}. ennreal (−ln (x∗y) ∗ x^(k+r) ∗ y^(k+s)) ∂lborel) = ...
qed
finally show ?thesis .
qed

Remembering that $\zeta(3) = \sum k^{-3}$, it is easy to see that if $r = s$, this sum is simply

$$2\left(\zeta(3) - \sum_{k=1}^{r} \frac{1}{k^3}\right) .$$

lemma beukers-nn-integral1-same:
assumes r = s
shows beukers-nn-integral1 r s = ennreal (2 ∗ (Re (zeta 3) − (∑ k=1..r. 1 / k ^ 3)))
and 2 ∗ (Re (zeta 3) − (∑ k=1..r. 1 / k ^ 3)) ≥ 0
proof −
from assms have [simp]: s = r by simp
have ∗: Suc 2 = 3 by simp
have beukers-nn-integral1 r s = (∑ k. ennreal (2 / (r + k + 1) ^ 3))
unfolding beukers-nn-integral1-series
by (simp only: assms power-Suc [symmetric] mult.commute[of x ^ 2 for x] ∗
times-divide-eq-right mult-1-right add-ac flip: mult-2)
also have ∗∗: (λk. 2 / (r + k + 1) ^ 3) sums
(2 ∗ (Re (zeta 3) − (∑ k=1..r. 1 / k ^ 3)))
using sums-mult[OF sums-Re-zeta-of-nat-offset[of 3], of 2] by simp

hence ($\sum k$. *ennreal* (*2* / (*r* + *k* + *1*) ^ *3*)) = *ennreal* ...
by (*intro suminf-ennreal-eq*) *auto*
finally show *beukers-nn-integral1 r s* = *ennreal* (*2* ∗ (*Re* (*zeta 3*) − ($\sum k$=*1*..*r*. *1* / *k* ^ *3*))) **.**
show *2* ∗ (*Re* (*zeta 3*) − ($\sum k$=*1*..*r*. *1* / *k* ^ *3*)) ≥ *0*
by (*rule sums-le*[*OF - sums-zero* ∗∗]) *auto*
qed

lemma *beukers-integral1-same*:
assumes *r* = *s*
shows *beukers-integral1 r s* = *2* ∗ (*Re* (*zeta 3*) − ($\sum k$=*1*..*r*. *1* / *k* ^ *3*))
proof −
have *ln* (*a* ∗ *b*) ∗ *a* ^ *r* ∗ *b* ^ *s* / (*1* − *a* ∗ *b*) ≤ *0* **if** *a* ∈ {*0*<..<*1*} *b* ∈ {*0*<..<*1*} **for** *a b* :: *real*
using *that mult-strict-mono*[*of a 1 b 1*] **by** (*intro mult-nonpos-nonneg divide-nonpos-nonneg*) *auto*
thus *?thesis*
using *beukers-nn-integral1-same*[*OF assms*]
unfolding *beukers-nn-integral1-def beukers-integral1-def*
by (*intro set-integral-eq-nn-integral AE-I2*)
(*auto simp flip*: *lborel-prod simp*: *case-prod-unfold set-borel-measurable-def*
intro: *divide-nonpos-nonneg mult-nonpos-nonneg*)
qed

In contrast, for $r > s$, we find that

$$I_1 = \frac{1}{r-s}\sum_{k=s+1}^{r}\frac{1}{k^2}\ .$$

lemma *beukers-nn-integral1-different*:
assumes *r* > *s*
shows *beukers-nn-integral1 r s* = *ennreal* (($\sum k$∈{*s*<..*r*}. *1* / *k* ^ *2*) / (*r* − *s*))
proof −
have (λ*k*. *1* / (*r* − *s*) ∗ (*1* / (*s* + *k* + *1*) ^ *2* − *1* / (*r* + *k* + *1*) ^ *2*))
sums (*1* / (*r* − *s*) ∗ ((*Re* (*zeta* (*of-nat 2*)) − ($\sum k$=*1*..*s*. *1*/*k*^*2*)) −
(*Re* (*zeta* (*of-nat 2*)) − ($\sum k$=*1*..*r*. *1*/*k*^*2*))))
(**is** - *sums ?S*) **by** (*intro sums-mult sums-diff sums-Re-zeta-of-nat-offset*) *auto*
also have *?S* = (($\sum k$=*1*..*r*. *1*/*k*^*2*) − ($\sum k$=*1*..*s*. *1*/*k*^*2*)) / (*r* − *s*)
by (*simp add*: *algebra-simps diff-divide-distrib*)
also have ($\sum k$=*1*..*r*. *1*/*k*^*2*) − ($\sum k$=*1*..*s*. *1*/*k*^*2*) = ($\sum k$∈{*1*..*r*}−{*1*..*s*}. *1*/*k*^*2*)
using *assms* **by** (*subst Groups-Big.sum-diff*) *auto*
also have {*1*..*r*} − {*1*..*s*} = {*s*<..*r*} **by** *auto*

also have (λ*k*. *1* / (*r* − *s*) ∗ (*1* / (*s* + *k* + *1*) ^ *2* − *1* / (*r* + *k* + *1*) ^ *2*)) =
(λ*k*. *1* / ((*k*+*r*+*1*) ∗ (*k*+*s*+*1*)^*2*) + *1* / ((*k*+*r*+*1*)^*2* ∗ (*k*+*s*+*1*)))
proof (*intro ext*, *goal-cases*)
case (*1 k*)
define *x* **where** *x* = *real* (*k* + *r* + *1*)

define *y* **where** *y* = *real* (*k* + *s* + *1*)
have [*simp*]: *x* ≠ *0* *y* ≠ *0* **by** (*auto simp*: *x-def y-def*)
have (x^2 ∗ *y* + *x* ∗ y^2) ∗ (*real r* − *real s*) = *x* ∗ *y* ∗ (x^2 − y^2)
by (*simp add*: *algebra-simps power2-eq-square x-def y-def*)
hence *1* / (*x*∗*y*^*2*) + *1* / (*x*^*2*∗*y*) = *1* / (*r* − *s*) ∗ (*1* / *y*^*2* − *1* / *x*^*2*)
using *assms* **by** (*simp add*: *divide-simps of-nat-diff*)
thus *?case* **by** (*simp add*: *x-def y-def algebra-simps*)
qed
finally show *?thesis*
unfolding *beukers-nn-integral1-series* **by** (*intro suminf-ennreal-eq*) (*auto simp*: *add-ac*)
qed

lemma *beukers-integral1-different*:
assumes *r* > *s*
shows *beukers-integral1 r s* = (∑ *k*∈{*s*<..*r*}. *1* / *k* ^ *2*) / (*r* − *s*)
proof −
have *ln* (*a* ∗ *b*) ∗ *a* ^ *r* ∗ *b* ^ *s* / (*1* − *a* ∗ *b*) ≤ *0* **if** *a* ∈ {*0*<..<*1*} *b* ∈ {*0*<..<*1*} **for** *a b* :: *real*
using *that mult-strict-mono*[*of a 1 b 1*] **by** (*intro mult-nonpos-nonneg divide-nonpos-nonneg*) *auto*
thus *?thesis*
using *beukers-nn-integral1-different*[*OF assms*]
unfolding *beukers-nn-integral1-def beukers-integral1-def*
by (*intro set-integral-eq-nn-integral AE-I2*)
(*auto simp flip*: *lborel-prod simp*: *case-prod-unfold set-borel-measurable-def intro*: *divide-nonpos-nonneg mult-nonpos-nonneg intro*!: *sum-nonneg divide-nonneg-nonneg*)
qed

end

It is also easy to see that if we exchange *r* and *s*, nothing changes.

lemma *beukers-nn-integral1-swap*:
beukers-nn-integral1 r s = *beukers-nn-integral1 s r*
unfolding *beukers-nn-integral1-def lborel-prod* [*symmetric*]
by (*subst lborel-pair.nn-integral-swap*, *simp*)
(*intro nn-integral-cong*, *auto simp*: *indicator-def algebra-simps split*: *if-splits*)

lemma *beukers-nn-integral1-finite*: *beukers-nn-integral1 r s* < ∞
using *beukers-nn-integral1-different*[*of r s*] *beukers-nn-integral1-different*[*of s r*]
by (*cases r s rule*: *linorder-cases*)
(*simp-all add*: *beukers-nn-integral1-same beukers-nn-integral1-swap*)

lemma *beukers-integral1-integrable*:
set-integrable lborel ({*0*<..<*1*}×{*0*<..<*1*})
(λ(*x*,*y*). (−*ln* (*x*∗*y*) / (*1* − *x*∗*y*) ∗ *x*^*r* ∗ *y*^*s* :: *real*))
proof (*intro set-integrableI-nonneg AE-I2*; *clarify?*)
fix *x y* :: *real* **assume** *xy*: *x* ∈ {*0*<..<*1*} *y* ∈ {*0*<..<*1*}

have *0 ≥ ln (x ∗ y) / (1 − x ∗ y) ∗ x ^ r ∗ y ^ s*
  using *mult-strict-mono[of x 1 y 1]*
  by (*intro mult-nonpos-nonneg divide-nonpos-nonneg*) (*use xy* **in** *auto*)
thus *0 ≤ − ln (x ∗ y) / (1 − x ∗ y) ∗ x ^ r ∗ y ^ s* **by** *simp*
next
show (∫⁺x∈{0<..<1}×{0<..<1}. *ennreal* (*case x of* (x, y) ⇒
  − ln (x ∗ y) / (1 − x ∗ y) ∗ x ^ r ∗ y ^ s) ∂*lborel*) < ∞
 using *beukers-nn-integral1-finite* **by** (*simp add*: *beukers-nn-integral1-def case-prod-unfold*)
qed (*simp-all flip*: *lborel-prod add*: *set-borel-measurable-def*)

lemma *beukers-integral1-integrable′*:
 set-integrable lborel ({0<..<1}×{0<..<1}×{0<..<1})
  (λ(z,x,y). (x^r ∗ y^s / (1 − (1 − x∗y) ∗ z) :: *real*))
proof (*intro set-integrableI-nonneg AE-I2*; *clarify?*)
 fix *x y z* :: *real* **assume** *xyz*: x ∈ {0<..<1} y ∈ {0<..<1} z ∈ {0<..<1}
 show *0 ≤ x^r ∗ y^s / (1 − (1 − x∗y) ∗ z)*
  using *mult-strict-mono[of x 1 y 1] xyz beukers-denom-ineq[of x y z]*
  by (*intro mult-nonneg-nonneg divide-nonneg-nonneg*) *auto*
next
 show (∫⁺x∈{0<..<1}×{0<..<1}×{0<..<1}. *ennreal* (*case x of* (z,x,y) ⇒
  x ^ r ∗ y ^ s / (1−(1−x∗y)∗z)) ∂*lborel*) < ∞
  using *beukers-nn-integral1-finite*
  by (*simp add*: *beukers-nn-integral1-altdef case-prod-unfold*)
qed (*simp-all flip*: *lborel-prod add*: *set-borel-measurable-def*)

lemma *beukers-integral1-conv-nn-integral*:
 beukers-integral1 r s = enn2real (*beukers-nn-integral1 r s*)
proof −
 have *ln (a ∗ b) ∗ a ^ r ∗ b ^ s / (1 − a ∗ b) ≤ 0* **if** a ∈ {0<..<1} b ∈ {0<..<1}
  for *a b* :: *real*
 using *mult-strict-mono[of a 1 b 1] that* **by** (*intro divide-nonpos-nonneg mult-nonpos-nonneg*)
auto
 thus *?thesis* **unfolding** *beukers-integral1-def* **using** *beukers-nn-integral1-finite[of
r s]*
  by (*intro set-integral-eq-nn-integral*)
   (*auto simp*: *case-prod-unfold beukers-nn-integral1-def
    set-borel-measurable-def simp flip*: *borel-prod
    intro*!: *AE-I2 intro*: *divide-nonpos-nonneg mult-nonpos-nonneg*)
qed

lemma *beukers-integral1-swap*: *beukers-integral1 r s = beukers-integral1 s r*
 by (*simp add*: *beukers-integral1-conv-nn-integral beukers-nn-integral1-swap*)

1.7 The second double integral

context
 fixes *n* :: *nat*
 fixes *D* :: (*real* × *real*) *set* **and** *D′* :: (*real* × *real* × *real*) *set*
 fixes *P* :: *real* ⇒ *real* **and** *Q* :: *nat* ⇒ *real* ⇒ *real*

defines *D ≡ {0<..<1} × {0<..<1}* **and** *D′ ≡ {0<..<1} × {0<..<1} × {0<..<1}*
defines *Q ≡ Gen-Shleg n* **and** *P ≡ Shleg n*
begin

The next integral to consider is the following variant of I_1:

$$I_2 := \int_0^1 \int_0^1 -\frac{\ln(xy)}{1-xy} P_n(x) P_n(y)\,\mathrm{d}x\,\mathrm{d}y \ .$$

definition *beukers-integral2 :: real* **where**
beukers-integral2 = (∫ (x,y)∈D. (−ln (x∗y) / (1−x∗y) ∗ P x ∗ P y) ∂lborel)

I_2 is simply a sum of integrals of type I_1, so using our results for I_1, we can write I_2 in the form $A\zeta(3) + \frac{B}{\mathrm{lcm}\{1...n\}^3}$ where A and B are integers and $A > 0$:

lemma *beukers-integral2-conv-int-combination*:
obtains *A B :: int* **where** *A > 0* **and**
beukers-integral2 = of-int A ∗ Re (zeta 3) + of-int B / of-nat (Lcm {1..n} ^ 3)
proof −
let *?I1 = (λi. (i, i)) ‘ {..n}*
let *?I2 = Set.filter (λ(i,j). i ≠ j) ({..n}×{..n})*
let *?I3 = Set.filter (λ(i,j). i < j) ({..n}×{..n})*
let *?I4 = Set.filter (λ(i,j). i > j) ({..n}×{..n})*
define *p* **where** *p = shleg-poly n*
define *I* **where** *I = (SIGMA i:{..n}. {1..i})*
define *J* **where** *J = (SIGMA (i,j):?I4. {j<..i})*
define *h* **where** *h = beukers-integral1*
define *A :: int* **where** *A = (∑ i≤n. 2 ∗ poly.coeff p i ^ 2)*
define *B1* **where** *B1 = (∑ (i,k)∈I. real-of-int (−2 ∗ coeff p i ^ 2) / real-of-int (k ^ 3))*
define *B2* **where** *B2 = (∑ ((i,j),k)∈J. real-of-int (2 ∗ coeff p i ∗ coeff p j) / real-of-int (k^2∗(i−j)))*
define *d* **where** *d = Lcm {1..n} ^ 3*

have [*simp*]: *h i j = h j i* **for** *i j*
by (*simp add*: *h-def beukers-integral1-swap*)

have *beukers-integral2 =*
(∫ (x,y)∈D. (∑ (i,j)∈{..n}×{..n}. coeff p i ∗ coeff p j ∗
−ln (x∗y) / (1−x∗y) ∗ x ^ i ∗ y ^ j) ∂lborel)
unfolding *beukers-integral2-def*
by (*subst sum.cartesian-product* [*symmetric*])
(*simp add*: *poly-altdef P-def Shleg-def mult-ac case-prod-unfold p-def*
sum-distrib-left sum-distrib-right sum-negf sum-divide-distrib)
also have *. . . = (∑ (i,j)∈{..n}×{..n}. coeff p i ∗ coeff p j ∗ h i j)*
unfolding *case-prod-unfold*
proof (*subst set-integral-sum*)

```
    fix ij :: nat × nat
    have set-integrable lborel D
          (λ(x,y). real-of-int (coeff p (fst ij) ∗ coeff p (snd ij)) ∗
                (−ln (x∗y) / (1−x∗y) ∗ x ^ fst ij ∗ y ^ snd ij))
      unfolding case-prod-unfold using beukers-integral1-integrable[of fst ij snd ij]
      by (intro set-integrable-mult-right) (auto simp: D-def case-prod-unfold)
    thus set-integrable lborel D
          (λpa. real-of-int (coeff p (fst ij) ∗ coeff p (snd ij)) ∗
                −ln (fst pa ∗ snd pa) / (1 − fst pa ∗ snd pa) ∗ fst pa ^ fst ij ∗ snd
pa ^ snd ij)
      by (simp add: mult-ac case-prod-unfold)
  qed (auto simp: beukers-integral1-def h-def case-prod-unfold mult.assoc D-def
            simp flip: set-integral-mult-right)
  also have ... = (∑(i,j)∈?I1∪?I2. coeff p i ∗ coeff p j ∗ h i j)
    by (intro sum.cong) auto
  also have ... = (∑(i,j)∈?I1. coeff p i ∗ coeff p j ∗ h i j) +
                  (∑(i,j)∈?I2. coeff p i ∗ coeff p j ∗ h i j)
    by (intro sum.union-disjoint) auto
  also have (∑(i,j)∈?I1. coeff p i ∗ coeff p j ∗ h i j) =
                (∑i≤n. coeff p i ^ 2 ∗ h i i)
   by (subst sum.reindex) (auto intro: inj-onI simp: case-prod-unfold power2-eq-square)
  also have ... = (∑i≤n. coeff p i ^ 2 ∗ 2 ∗ (Re (zeta 3) − (∑k=1..i. 1 / k ^
3)))
    unfolding h-def D-def
    by (intro sum.cong refl, subst beukers-integral1-same) auto
  also have ... = of-int A ∗ Re (zeta 3) −
                    (∑i≤n. 2 ∗ coeff p i ^ 2 ∗ (∑k=1..i. 1 / k ^ 3))
     by (simp add: sum-subtractf sum-distrib-left sum-distrib-right algebra-simps
A-def)
  also have ... = of-int A ∗ Re (zeta 3) + B1
   unfolding I-def B1-def by (subst sum.Sigma [symmetric]) (auto simp: sum-distrib-left
sum-negf)
  also have (∑(i,j)∈?I2. coeff p i ∗ coeff p j ∗ h i j) =
                (∑(i,j)∈?I3∪?I4. coeff p i ∗ coeff p j ∗ h i j)
    by (intro sum.cong) auto
  also have ... = (∑(i,j)∈?I3. coeff p i ∗ coeff p j ∗ h i j) +
                  (∑(i,j)∈?I4. coeff p i ∗ coeff p j ∗ h i j)
    by (intro sum.union-disjoint) auto
  also have (∑(i,j)∈?I3. coeff p i ∗ coeff p j ∗ h i j) =
                (∑(i,j)∈?I4. coeff p i ∗ coeff p j ∗ h i j)
    by (intro sum.reindex-bij-witness[of - λ(i,j). (j,i) λ(i,j). (j,i)]) auto
  also have ... + ... = 2 ∗ ... by simp
  also have ... = (∑(i,j)∈?I4. ∑k∈{j<..i}. 2 ∗ coeff p i ∗ coeff p j / k ^ 2 /
(i − j))
    unfolding sum-distrib-left
    by (intro sum.cong refl)
      (auto simp: h-def beukers-integral1-different sum-divide-distrib sum-distrib-left
mult-ac)
  also have ... = B2
```

unfolding *J-def B2-def* **by** (*subst sum.Sigma* [*symmetric*]) (*auto simp*: *case-prod-unfold*)

also have ∃ *B1′*. *B1* = *real-of-int B1′* / *real-of-int d*
unfolding *B1-def case-prod-unfold*
by (*rule sum-rationals-common-divisor′*) (*auto simp*: *d-def I-def*)
then obtain *B1′* **where** *B1* = *real-of-int B1′* / *real-of-int d* **..**

also have ∃ *B2′*. *B2* = *real-of-int B2′* / *real-of-int d*
unfolding *B2-def case-prod-unfold J-def*
proof (*rule sum-rationals-common-divisor′*; *clarsimp?*)
fix *i j k* :: *nat* **assume** *ijk*: *i* ≤ *n j* < *k k* ≤ *i*
have *int* (*k* ^ *2* ∗ (*i* − *j*)) *dvd int* (*Lcm* {*1..n*} ^ *2* ∗ *Lcm* {*1..n*})
unfolding *int-dvd-int-iff* **using** *ijk*
by (*intro mult-dvd-mono dvd-power-same dvd-Lcm*) *auto*
also have . . . = *d*
by (*simp add*: *d-def power-numeral-reduce*)
finally show (*int k*)^2 ∗ (*int i* − *int j*) *dvd int d*
using *ijk* **by** *force*
qed(*auto simp*: *d-def J-def intro*!: *Nat.gr0I*)
then obtain *B2′* **where** *B2* = *real-of-int B2′* / *real-of-int d* **..**

finally have *beukers-integral2* =
of-int A ∗ *Re* (*zeta 3*) + *of-int* (*B1′* + *B2′*) / *of-nat* (*Lcm* {*1..n*}
^ *3*)
by (*simp add*: *add-divide-distrib d-def*)

moreover have *coeff p 0* = *P 0*
unfolding *P-def p-def Shleg-def* **by** (*simp add*: *poly-0-coeff-0*)
hence *coeff p 0* = *1*
by (*simp add*: *P-def*)
hence *A* > *0*
unfolding *A-def* **by** (*intro sum-pos2*[*of - 0*]) *auto*

ultimately show *?thesis*
by (*intro that*[*of A B1′* + *B2′*]) *auto*
qed

lemma *beukers-integral2-integrable*:
set-integrable lborel D (λ(*x*,*y*). −*ln* (*x*∗*y*) / (*1* − *x*∗*y*) ∗ *P x* ∗ *P y*)
proof −
have *bounded* (*P* ‘ {*0..1*})
unfolding *P-def Shleg-def*
by (*intro compact-imp-bounded compact-continuous-image continuous-intros*)
auto
then obtain *C* **where** *C*: ⋀*x*. *x* ∈ {*0..1*} ⟹ *norm* (*P x*) ≤ *C*
unfolding *bounded-iff* **by** *fast*
have [*measurable*]: *P* ∈ *borel-measurable borel* **by** (*simp add*: *P-def*)
from *C*[*of 0*] **have** *C* ≥ *0* **by** *simp*
show *?thesis*

proof (*rule set-integrable-bound*[*OF - - AE-I2*]; *clarify?*)
  show *set-integrable lborel D* (λ(*x*,*y*). *C* ^ *2* ∗ (−*ln* (*x*∗*y*) / (*1* − *x*∗*y*)))
    using *beukers-integral1-integrable*[*of 0 0*] **unfolding** *case-prod-unfold*
    by (*intro set-integrable-mult-right*) (*auto simp*: *D-def*)
next
  fix *x y* :: *real*
  assume *xy*: (*x*, *y*) ∈ *D*
  from *xy* **have** *x* ∗ *y* < *1*
    using *mult-strict-mono*[*of x 1 y 1*] **by** (*simp add*: *D-def*)
  have *norm* (−*ln* (*x*∗*y*) / (*1* − *x*∗*y*) ∗ *P x* ∗ *P y*) = (−*ln* (*x*∗*y*)) / (*1* − *x*∗*y*) ∗ *norm* (*P x*) ∗ *norm* (*P y*)
    using *xy* ‹*x* ∗ *y* < *1*› **by** (*simp add*: *abs-mult abs-divide D-def*)
  also have . . . ≤ (−*ln* (*x*∗*y*)) / (*1*−*x*∗*y*) ∗ *C* ∗ *C*
    using *xy C*[*of x*] *C*[*of y*] ‹*x* ∗ *y* < *1*› ‹*C* ≥ *0*›
    by (*intro mult-mono divide-left-mono*)
      (*auto simp*: *D-def divide-nonpos-nonneg mult-nonpos-nonneg*)
  also have . . . = *norm* ((−*ln* (*x*∗*y*)) / (*1*−*x*∗*y*) ∗ *C* ∗ *C*)
    using *xy* ‹*x* ∗ *y* < *1*› ‹*C* ≥ *0*› **by** (*simp add*: *abs-divide abs-mult D-def*)
  finally show *norm* (−*ln* (*x*∗*y*) / (*1* − *x*∗*y*) ∗ *P x* ∗ *P y*)
      ≤ *norm* (*case* (*x*, *y*) *of* (*x*, *y*) ⇒ C^2 ∗ (− *ln* (*x* ∗ *y*) / (*1* − *x* ∗ *y*)))
    by (*auto simp*: *algebra-simps power2-eq-square abs-mult abs-divide*)
qed (*auto simp*: *D-def set-borel-measurable-def case-prod-unfold simp flip*: *lborel-prod*)
qed

1.8 The triple integral

Lastly, we turn to the triple integral

$$I_3 := \int_0^1 \int_0^1 \int_0^1 \frac{(x(1-x)y(1-y)w(1-w))^n}{(1-(1-xy)w)^{n+1}}\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}w\ .$$

definition *beukers-nn-integral3* :: *ennreal* **where**
  beukers-nn-integral3 =
    (∫⁺(*w*,*x*,*y*)∈*D*′. ((*x*∗(*1*−*x*)∗*y*∗(*1*−*y*)∗*w*∗(*1*−*w*))^*n* / (*1*−(*1*−*x*∗*y*)∗*w*)^(*n*+*1*)) ∂*lborel*)

definition *beukers-integral3* :: *real* **where**
  beukers-integral3 =
    (∫ (*w*,*x*,*y*)∈*D*′. ((*x*∗(*1*−*x*)∗*y*∗(*1*−*y*)∗*w*∗(*1*−*w*))^*n* / (*1*−(*1*−*x*∗*y*)∗*w*)^(*n*+*1*)) ∂*lborel*)

We first prove the following bound (which is a consequence of the arithmetic–geometric mean inequality) that will help us bound the triple integral.

lemma *beukers-integral3-integrand-bound*:
  fixes *x y z* :: *real*
  assumes *xyz*: *x* ∈ {*0*<..<*1*} *y* ∈ {*0*<..<*1*} *z* ∈ {*0*<..<*1*}
  shows (*x*∗(*1*−*x*)∗*y*∗(*1*−*y*)∗*z*∗(*1*−*z*)) / (*1*−(*1*−*x*∗*y*)∗*z*) ≤ *1* / *27* (**is** *?lhs* ≤ -)
proof −
  have *ineq1*: *x* ∗ (*1* − *x*) ≤ *1* / *4* **if** *x*: *x* ∈ {*0*..*1*} **for** *x* :: *real*

```
  proof -
    have x * (1 - x) - 1 / 4 = -((x - 1 / 2) ^ 2)
      by (simp add: algebra-simps power2-eq-square)
    also have ... ≤ 0
      by simp
    finally show ?thesis by simp
  qed

  have ineq2: x * (1 - x) ^ 2 ≤ 4 / 27 if x: x ∈ {0..1} for x :: real
  proof -
    have x * (1 - x) ^ 2 - 4 / 27 = (x - 4 / 3) * (x - 1 / 3) ^ 2
      by (simp add: algebra-simps power2-eq-square)
    also have ... ≤ 0
      by (rule mult-nonpos-nonneg) (use x in auto)
    finally show ?thesis by simp
  qed

  have 1 - (1-x*y)*z = (1 - z) + x * y * z
    by (simp add: algebra-simps)
  also have ... ≥ 2 * sqrt (1 - z) * sqrt x * sqrt y * sqrt z
    using arith-geo-mean-sqrt[of 1 - z x * y * z] xyz
    by (auto simp: real-sqrt-mult)

  finally have *: ?lhs ≤ (x*(1-x)*y*(1-y)*z*(1-z)) / (2 * sqrt (1 - z) * sqrt
x * sqrt y * sqrt z)
    using xyz beukers-denom-ineq[of x y z]
    by (intro divide-left-mono mult-nonneg-nonneg mult-pos-pos) auto

  have (x*(1-x)*y*(1-y)*z*(1-z)) = (sqrt x * sqrt x * (1-x) * sqrt y * sqrt y
*
               (1-y) * sqrt z * sqrt z * sqrt (1-z) * sqrt (1-z))
    using xyz by simp
  also have ... / (2 * sqrt (1 - z) * sqrt x * sqrt y * sqrt z) =
             sqrt (x * (1 - x) ^ 2) * sqrt (y * (1 - y) ^ 2) * sqrt (z * (1 - z)) / 2
    using xyz by (simp add: divide-simps real-sqrt-mult del: real-sqrt-mult-self)
  also have ... ≤ sqrt (4 / 27) * sqrt (4 / 27) * sqrt (1 / 4) / 2
   using xyz by (intro divide-right-mono mult-mono real-sqrt-le-mono ineq1 ineq2)
auto
  also have ... = 1 / 27
    by (simp add: real-sqrt-divide)
  finally show ?thesis using * by argo
qed
```

Connecting the above bound with our results of I_1, it is easy to see that $I_3 \leq 2 \cdot 27^{-n} \cdot \zeta(3)$:

```
lemma beukers-nn-integral3-le:
  beukers-nn-integral3 ≤ ennreal (2 * (1 / 27) ^ n * Re (zeta 3))
proof -
  have D' [measurable]: D' ∈ sets (borel ⨂M borel ⨂M borel)
```

unfolding *D′-def* **by** (*simp flip*: *borel-prod*)
have *beukers-nn-integral3* =
(∫⁺(w,x,y)∈D′. ((x∗(1−x)∗y∗(1−y)∗w∗(1−w))^n / (1−(1−x∗y)∗w)^(n+1)) ∂lborel)
by (*simp add*: *beukers-nn-integral3-def*)
also have ... ≤ (∫⁺(w,x,y)∈D′. ((1 / 27) ^ n / (1−(1−x∗y)∗w)) ∂lborel)
proof (*intro set-nn-integral-mono ennreal-leI*, *clarify*, *goal-cases*)
case (*1 w x y*)
have (x∗(1−x)∗y∗(1−y)∗w∗(1−w))^n / (1−(1−x∗y)∗w)^(n+1) =
((x∗(1−x)∗y∗(1−y)∗w∗(1−w)) / (1−(1−x∗y)∗w)) ^ n / (1−(1−x∗y)∗w)
by (*simp add*: *divide-simps*)
also have ... ≤ (1 / 27) ^ n / (1 − (1 − x ∗ y) ∗ w)
using *beukers-denom-ineq*[*of x y w*] *1*
by (*intro divide-right-mono power-mono beukers-integral3-integrand-bound*)
(*auto simp*: *D′-def*)
finally show *?case* **.**
qed
also have ... = *ennreal* ((1 / 27) ^ n) ∗ (∫⁺(w,x,y)∈D′. (1 / (1−(1−x∗y)∗w)) ∂lborel)
unfolding *lborel-prod* [*symmetric*] *case-prod-unfold*
by (*subst nn-integral-cmult* [*symmetric*])
(*auto intro*!: *nn-integral-cong simp*: *indicator-def simp flip*: *ennreal-mult′*)
also have (∫⁺(w,x,y)∈D′. (1 / (1−(1−x∗y)∗w)) ∂lborel) =
(∫⁺(x,y)∈{0<..<1}×{0<..<1}. *ennreal* (− (*ln* (x ∗ y) / (1 − x ∗ y))) ∂lborel)
using *beukers-nn-integral1-altdef*[*of 0 0*]
by (*simp add*: *beukers-nn-integral1-def D′-def case-prod-unfold*)
also have ... = *ennreal* (2 ∗ *Re* (*zeta 3*))
using *beukers-nn-integral1-same*[*of 0 0*] **by** (*simp add*: *D-def beukers-nn-integral1-def*)
also have *ennreal* ((1 / 27) ^ n) ∗ ... = *ennreal* (2 ∗ (1 / 27) ^ n ∗ *Re* (*zeta 3*))
by (*subst ennreal-mult′* [*symmetric*]) (*simp-all add*: *mult-ac*)
finally show *?thesis* **.**
qed

lemma *beukers-nn-integral3-finite*: *beukers-nn-integral3* < ∞
by (*rule le-less-trans*, *rule beukers-nn-integral3-le*) *simp-all*

lemma *beukers-integral3-integrable*:
set-integrable lborel D′ (λ(w,x,y). (x∗(1−x)∗y∗(1−y)∗w∗(1−w))^n / (1−(1−x∗y)∗w)^(n+1))
unfolding *case-prod-unfold* **using** *less-imp-le*[*OF beukers-denom-ineq*] *beukers-nn-integral3-finite*
by (*intro set-integrableI-nonneg AE-I2 impI*)
(*auto simp*: *D′-def set-borel-measurable-def beukers-nn-integral3-def case-prod-unfold*
simp flip: *lborel-prod intro*!: *divide-nonneg-nonneg mult-nonneg-nonneg*)

lemma *beukers-integral3-conv-nn-integral*:
beukers-integral3 = *enn2real beukers-nn-integral3*
unfolding *beukers-integral3-def* **using** *beukers-nn-integral3-finite less-imp-le*[*OF beukers-denom-ineq*]

```
  by (intro set-integral-eq-nn-integral AE-I2 impI)
    (auto simp: D'-def set-borel-measurable-def beukers-nn-integral3-def case-prod-unfold
          simp flip: lborel-prod)

lemma beukers-integral3-le: beukers-integral3 ≤ 2 * (1 / 27) ^ n * Re (zeta 3)
proof -
  have beukers-integral3 = enn2real beukers-nn-integral3
    by (rule beukers-integral3-conv-nn-integral)
  also have ... ≤ enn2real (ennreal (2 * (1 / 27) ^ n * Re (zeta 3)))
    by (intro enn2real-mono beukers-nn-integral3-le) auto
  also have ... = 2 * (1 / 27) ^ n * Re (zeta 3)
    using Re-zeta-ge-1[of 3] by (intro enn2real-ennreal mult-nonneg-nonneg) auto
  finally show ?thesis .
qed
```

It is also easy to see that $I_3 > 0$.

```
lemma beukers-nn-integral3-pos: beukers-nn-integral3 > 0
proof -
  have D' [measurable]: D' ∈ sets (borel ⨂M borel ⨂M borel)
    unfolding D'-def by (simp flip: borel-prod)

  have *: ¬(AE (w,x,y) in lborel. ennreal ((x*(1−x)*y*(1−y)*w*(1−w))^n /
            (1−(1−x*y)*w)^(n+1)) * indicator D' (w,x,y) ≤ 0)
          (is ¬(AE z in lborel. ?P z))
  proof -
    {
      fix w x y :: real assume xyw: (w,x,y) ∈ D'
      hence (x*(1−x)*y*(1−y)*w*(1−w))^n / (1−(1−x*y)*w)^(n+1) > 0
        using beukers-denom-ineq[of x y w]
        by (intro divide-pos-pos mult-pos-pos zero-less-power) (auto simp: D'-def)
      with xyw have ¬?P (w,x,y)
        by (auto simp: indicator-def D'-def)
    }
    hence *: ¬?P z if z ∈ D' for z using that by blast
    hence {z∈space lborel. ¬?P z} = D' by auto
    moreover have emeasure lborel D' = 1
    proof -
      have D' = box (0,0,0) (1,1,1)
        by (auto simp: D'-def box-def Basis-prod-def)
      also have emeasure lborel ... = 1
        by (subst emeasure-lborel-box) (auto simp: Basis-prod-def)
      finally show ?thesis by simp
    qed
    ultimately show ?thesis
      by (subst AE-iff-measurable[of D']) (simp-all flip: borel-prod)
  qed

  hence nn-integral lborel (λ-::real×real×real. 0) < beukers-nn-integral3
    unfolding beukers-nn-integral3-def
```

```
    by (intro nn-integral-less) (simp-all add: case-prod-unfold flip: lborel-prod)
  thus ?thesis by simp
qed

lemma beukers-integral3-pos: beukers-integral3 > 0
proof −
  have 0 < enn2real beukers-nn-integral3
    using beukers-nn-integral3-pos beukers-nn-integral3-finite
    by (subst enn2real-positive-iff) auto
  also have ... = beukers-integral3
    by (rule beukers-integral3-conv-nn-integral [symmetric])
  finally show ?thesis .
qed
```

1.9 Connecting the double and triple integral

In this section, we will prove the most technically involved part, namely that $I_2 = I_3$. I will not go into detail about how this works – the reader is advised to simply look at Filaseta's presentation of the proof.

The basic idea is to integrate by parts n times with respect to y to eliminate the factor $P(y)$, then change variables $z = \frac{1-w}{1-(1-xy)w}$, and then apply the same integration by parts n times to x to eliminate $P(x)$.

The first expand

$$-\frac{\ln(xy)}{1-xy} = \int_0^1 \frac{1}{1-(1-xy)z}\,\mathrm{d}z\ .$$

```
lemma beukers-aux-ln-conv-integral:
  fixes x y :: real
  assumes xy: x ∈ {0<..<1} y ∈ {0<..<1}
  shows −ln (x∗y) / (1−x∗y) = (LBINT z=0..1. 1 / (1−(1−x∗y)∗z))
proof −
  have x ∗ y < 1
    using mult-strict-mono[of x 1 y 1] xy by simp
  have less: (1 − x ∗ y) ∗ u < 1 if u: u ∈ {0..1} for u
  proof −
    from u ‹x ∗ y < 1› have (1 − x ∗ y) ∗ u ≤ (1 − x ∗ y) ∗ 1
      by (intro mult-left-mono) auto
    also have ... < 1 ∗ 1
      using xy by (intro mult-strict-right-mono) auto
    finally show (1 − x ∗ y) ∗ u < 1 by simp
  qed
  have neq: (1 − x ∗ y) ∗ u ≠ 1 if u ∈ {0..1} for u
    using less[of u] that by simp

  let ?F = λz. ln (1−(1−x∗y)∗z)/(x∗y−1)
  have (LBINT z=ereal 0..ereal 1. 1 / (1−(1−x∗y)∗z)) = ?F 1 − ?F 0
```

```
  proof (rule interval-integral-FTC-finite, goal-cases cont deriv)
    case cont
    show ?case
      using neq by (intro continuous-intros) auto
  next
    case (deriv z)
    show ?case
      unfolding has-real-derivative-iff-has-vector-derivative [symmetric]
      by (insert less[of z] xy ‹x ∗ y < 1› deriv)
         (rule derivative-eq-intros refl | simp)+
  qed
  also have . . . = −ln (x∗y) / (1−x∗y)
    using ‹x ∗ y < 1› by (simp add: field-simps)
  finally show ?thesis
    by (simp add: zero-ereal-def one-ereal-def)
qed
```

The first part we shall show is the integration by parts.

```
lemma beukers-aux-by-parts-aux:
  assumes xz: x ∈ {0<..<1} z ∈ {0<..<1} and k ≤ n
  shows (LBINT y=0..1. Q n y ∗ (1/(1−(1−x∗y)∗z))) =
        (LBINT y=0..1. Q (n−k) y ∗ (fact k ∗ (x∗z)^k / (1−(1−x∗y)∗z) ^(k+1)))
  using assms(3)
proof (induction k)
  case (Suc k)
  note [derivative-intros] = DERIV-chain2[OF has-field-derivative-Gen-Shleg]
  define G where G = (λy. −fact k ∗ (x∗z)^k / (1−(1−x∗y)∗z) ^(k+1))
  define g where g = (λy. fact (Suc k) ∗ (x∗z)^Suc k / (1−(1−x∗y)∗z) ^(k+2))

  have less: (1 − x ∗ y) ∗ z < 1 and neq: (1 − x ∗ y) ∗ z ≠ 1
    if y: y ∈ {0..1} for y
  proof −
    from y xz have x ∗ y ≤ x ∗ 1
      by (intro mult-left-mono) auto
    also have . . . < 1
      using xz by simp
    finally have (1 − x ∗ y) ∗ z ≤ 1 ∗ z
      using xz y by (intro mult-right-mono) auto
    also have . . . < 1
      using xz by simp
    finally show (1 − x ∗ y) ∗ z < 1 by simp
    thus (1 − x ∗ y) ∗ z ≠ 1 by simp
  qed

  have cont: continuous-on {0..1} g
    using neq by (auto simp: g-def intro!: continuous-intros)
  have deriv: (G has-real-derivative g y) (at y within {0..1}) if y: y ∈ {0..1} for
y
    unfolding G-def
```

by (*insert neq xz y*, (*rule derivative-eq-intros refl power-not-zero*)+)
(*auto simp*: *divide-simps g-def*)
have *deriv2*: (*Q* (*n* − *Suc k*) *has-real-derivative Q* (*n* − *k*) *y*) (*at y within* {*0..1*})
for *y*
using *Suc.prems* **by** (*auto intro*!: *derivative-eq-intros simp*: *Suc-diff-Suc Q-def*)

have (*LBINT y=0..1. Q* (*n*−*Suc k*) *y* ∗ (*fact* (*Suc k*) ∗ (*x*∗*z*)^*Suc k* / (*1*−(*1*−*x*∗*y*)∗*z*)
^(*k*+*2*))) =
(*LBINT y=0..1. Q* (*n*−*Suc k*) *y* ∗ *g y*)
by (*simp add*: *g-def*)
also have (*LBINT y=0..1. Q* (*n*−*Suc k*) *y* ∗ *g y*) = −(*LBINT y=0..1. Q* (*n*−*k*)
y ∗ *G y*)
using *Suc.prems deriv deriv2 cont*
by (*subst interval-lebesgue-integral-by-parts-01*[**where** *f* = *Q* (*n*−*k*) **and** *G* =
G])
(*auto intro*!: *continuous-intros simp*: *Q-def*)
also have ... = (*LBINT y=0..1. Q* (*n*−*k*) *y* ∗ (*fact k* ∗ (*x*∗*z*)^*k* / (*1*−(*1*−*x*∗*y*)∗*z*)
^(*k*+*1*)))
by (*simp add*: *G-def flip*: *interval-lebesgue-integral-uminus*)
finally show *?case* **using** *Suc* **by** *simp*
qed *auto*

lemma *beukers-aux-by-parts*:
assumes *xz*: *x* ∈ {*0*<..<*1*} *z* ∈ {*0*<..<*1*}
shows (*LBINT y=0..1. P y* / (*1*−(*1*−*x*∗*y*)∗*z*)) =
(*LBINT y=0..1.* (*x*∗*y*∗*z*)^*n* ∗ (*1*−*y*)^*n* / (*1*−(*1*−*x*∗*y*)∗*z*) ^(*n*+*1*))
proof −
have (*LBINT y=0..1. P y* ∗ (*1*/(*1*−(*1*−*x*∗*y*)∗*z*))) =
1 / *fact n* ∗ (*LBINT y=0..1. Q n y* ∗ (*1*/(*1*−(*1*−*x*∗*y*)∗*z*)))
unfolding *interval-lebesgue-integral-mult-right* [*symmetric*]
by (*simp add*: *P-def Q-def Shleg-altdef*)
also have ... = (*LBINT y=0..1.* (*x*∗*y*∗*z*)^*n* ∗ (*1*−*y*)^*n* / (*1*−(*1*−*x*∗*y*)∗*z*) ^
(*n*+*1*))
by (*subst beukers-aux-by-parts-aux*[*OF assms*, *of n*], *simp*,
subst interval-lebesgue-integral-mult-right [*symmetric*])
(*simp add*: *Q-def mult-ac Gen-Shleg-0-left power-mult-distrib*)
finally show *?thesis* **by** *simp*
qed

We then get the following by applying the integration by parts to *y*:

lemma *beukers-aux-integral-transform1*:
fixes *z* :: *real*
assumes *z*: *z* ∈ {*0*<..<*1*}
shows (*LBINT* (*x*,*y*):*D. P x* ∗ *P y* / (*1*−(*1*−*x*∗*y*)∗*z*)) =
(*LBINT* (*x*,*y*):*D. P x* ∗ (*x*∗*y*∗*z*)^*n* ∗ (*1*−*y*)^*n* / (*1*−(*1*−*x*∗*y*)∗*z*)^(*n*+*1*))
proof −
have *cbox*: *cbox* (*0*, *0*) (*1*, *1*) = ({*0..1*} × {*0..1*} :: (*real* × *real*) *set*)
by (*auto simp*: *cbox-def Basis-prod-def inner-prod-def*)
have *box*: *box* (*0*, *0*) (*1*, *1*) = ({*0*<..<*1*} × {*0*<..<*1*} :: (*real* × *real*) *set*)

```
    by (auto simp: box-def Basis-prod-def inner-prod-def)
  have set-integrable lborel (cbox (0,0) (1,1))
          (λ(x, y). P x ∗ P y / (1 − (1 − x ∗ y) ∗ z))
    unfolding lborel-prod case-prod-unfold P-def
  proof (intro continuous-on-imp-set-integrable-cbox continuous-intros ballI)
    fix p :: real × real assume p: p ∈ cbox (0, 0) (1, 1)
    have (1 − fst p ∗ snd p) ∗ z ≤ 1 ∗ z
      using mult-mono[of fst p 1 snd p 1] p z cbox by (intro mult-right-mono) auto
    also have ... < 1 using z by simp
    finally show 1 − (1 − fst p ∗ snd p) ∗ z ≠ 0 by simp
  qed
  hence integrable: set-integrable lborel (box (0,0) (1,1))
          (λ(x, y). P x ∗ P y / (1 − (1 − x ∗ y) ∗ z))
    by (rule set-integrable-subset) (auto simp: box simp flip: borel-prod)

  have set-integrable lborel (cbox (0,0) (1,1))
          (λ(x, y). P x ∗ (x∗y∗z)^n ∗ (1−y)^n / (1−(1−x∗y)∗z)^(n+1))
    unfolding lborel-prod case-prod-unfold P-def
  proof (intro continuous-on-imp-set-integrable-cbox continuous-intros ballI)
    fix p :: real × real assume p: p ∈ cbox (0, 0) (1, 1)
    have (1 − fst p ∗ snd p) ∗ z ≤ 1 ∗ z
      using mult-mono[of fst p 1 snd p 1] p z cbox by (intro mult-right-mono) auto
    also have ... < 1 using z by simp
    finally show (1 − (1 − fst p ∗ snd p) ∗ z) ^ (n + 1) ≠ 0 by simp
  qed
  hence integrable': set-integrable lborel D
          (λ(x, y). P x ∗ (x∗y∗z)^n ∗ (1−y)^n / (1−(1−x∗y)∗z)^(n+1))
    by (rule set-integrable-subset) (auto simp: box D-def simp flip: borel-prod)

  have (LBINT (x,y):D. P x ∗ P y / (1−(1−x∗y)∗z)) =
          (LBINT x=0..1. (LBINT y=0..1. P x ∗ P y / (1−(1−x∗y)∗z)))
    unfolding D-def lborel-prod [symmetric] using box integrable
   by (subst lborel-pair.set-integral-fst') (simp-all add: interval-integral-Ioo lborel-prod)
  also have ... = (LBINT x=0..1. P x ∗ (LBINT y=0..1. P y / (1−(1−x∗y)∗z)))
    by (subst interval-lebesgue-integral-mult-right [symmetric]) (simp add: mult-ac)
  also have ... = (LBINT x=0..1. P x ∗ (LBINT y=0..1. (x∗y∗z)^n ∗ (1−y)^n
/ (1−(1−x∗y)∗z)^(n+1)))
   using z by (intro interval-lebesgue-integral-lborel-01-cong, subst beukers-aux-by-parts)
auto
  also have ... = (LBINT x=0..1. (LBINT y=0..1. P x ∗ (x∗y∗z)^n ∗ (1−y)^n
/ (1−(1−x∗y)∗z)^(n+1)))
    by (subst interval-lebesgue-integral-mult-right [symmetric]) (simp add: mult-ac)
  also have ... = (LBINT (x,y):D. P x ∗ (x∗y∗z)^n ∗ (1−y)^n / (1−(1−x∗y)∗z)^(n+1))
    unfolding D-def lborel-prod [symmetric] using box integrable'
    by (subst lborel-pair.set-integral-fst')
       (simp-all add: D-def interval-integral-Ioo lborel-prod)
  finally show (LBINT (x,y):D. P x ∗ P y / (1−(1−x∗y)∗z)) = ... .
qed
```

We then change variables for z:

```
lemma beukers-aux-integral-transform2:
  assumes xy: x ∈ {0<..<1} y ∈ {0<..<1}
  shows (LBINT z=0..1. (x∗y∗z)^n ∗ (1−y)^n / (1−(1−x∗y)∗z)^(n+1)) =
         (LBINT w=0..1. (1−w)^n ∗ (1−y)^n / (1−(1−x∗y)∗w))
proof −
  define g where g = (λz. (1 − z) / (1−(1−x∗y)∗z))
  define g′ where g′ = (λz. −x∗y / (1−(1−x∗y)∗z)^2)
  have x ∗ y < 1
    using mult-strict-mono[of x 1 y 1] xy by simp
  have less: (1 − (x ∗ y)) ∗ w < 1 and neq: (1 − (x ∗ y)) ∗ w ≠ 1
    if w: w ∈ {0..1} for w
  proof −
    have (1 − (x ∗ y)) ∗ w ≤ (1 − (x ∗ y)) ∗ 1
      using w ‹x ∗ y < 1› by (intro mult-left-mono) auto
    also have . . . < 1
      using xy by simp
    finally show (1 − (x ∗ y)) ∗ w < 1 .
    thus (1 − (x ∗ y)) ∗ w ≠ 1 by simp
  qed

  have deriv: (g has-real-derivative g′ w) (at w within {0..1}) if w ∈ {0..1} for w
    unfolding g-def g′-def
    apply (insert that xy neq)
    apply (rule derivative-eq-intros refl)+
     apply (simp-all add: divide-simps power2-eq-square)
     apply (auto simp: algebra-simps)
    done
  have continuous-on {0..1} (λxa. (1 − xa) ^ n / (1 − (1 − x ∗ y) ∗ xa))
    using neq by (auto intro!: continuous-intros)
  moreover have g ‘ {0..1} ⊆ {0..1}
  proof clarify
    fix w :: real assume w: w ∈ {0..1}
    have (1 − x ∗ y) ∗ w ≤ 1 ∗ w
      using ‹x ∗ y < 1› xy w by (intro mult-right-mono) auto
    thus g w ∈ {0..1}
      unfolding g-def using less[of w] w by (auto simp: divide-simps)
  qed
  ultimately have cont: continuous-on (g ‘ {0..1}) (λxa. (1 − xa) ^ n / (1 − (1
− x ∗ y) ∗ xa))
    by (rule continuous-on-subset)
  have cont′: continuous-on {0..1} g′
    using neq by (auto simp: g′-def intro!: continuous-intros)

  have (LBINT w=0..1. (1−w)^n ∗ (1−y)^n / (1−(1−x∗y)∗w)) =
          (1−y)^n ∗ (LBINT w=0..1. (1 − w)^n / (1−(1−x∗y)∗w))
    unfolding interval-lebesgue-integral-mult-right [symmetric]
    by (simp add: algebra-simps power-mult-distrib)
  also have (LBINT w=0..1. (1 − w)^n / (1−(1−x∗y)∗w)) =
         −(LBINT w=g 0..g 1. (1 − w)^n / (1−(1−x∗y)∗w))
```

```
    by (subst interval-integral-endpoints-reverse)(simp add: g-def zero-ereal-def
one-ereal-def)
  also have (LBINT w=g 0..g 1. (1 − w)^n / (1−(1−x∗y)∗w)) =
             (LBINT w=0..1. g′ w ∗ ((1 − g w) ^ n / (1 − (1−x∗y) ∗ g w)))
    using deriv cont cont′
    by (subst interval-integral-substitution-finite [symmetric, where g = g and g′
= g′])
      (simp-all add: zero-ereal-def one-ereal-def)
  also have −... = (LBINT z=0..1. ((x∗y)^n ∗ z^n / (1−(1−x∗y)∗z)^(n+1)))
    unfolding interval-lebesgue-integral-uminus [symmetric] using xy
    apply (intro interval-lebesgue-integral-lborel-01-cong)
    apply (simp add: divide-simps g-def g′-def)
    apply (auto simp: algebra-simps power-mult-distrib power2-eq-square)
    done
  also have (1−y)^n ∗ ... = (LBINT z=0..1. (x∗y∗z)^n ∗ (1−y)^n / (1−(1−x∗y)∗z)^(n+1))
    unfolding interval-lebesgue-integral-mult-right [symmetric]
    by (simp add: algebra-simps power-mult-distrib)
  finally show ... = (LBINT w=0..1. (1−w)^n ∗ (1−y)^n / (1−(1−x∗y)∗w))
..
qed
```

Lastly, we apply the same integration by parts to x:

```
lemma beukers-aux-integral-transform3:
  assumes w: w ∈ {0<..<1}
  shows   (LBINT (x,y):D. P x ∗ (1−y)^n / (1−(1−x∗y)∗w)) =
        (LBINT (x,y):D. (1−y)^n ∗ (x∗y∗w)^n ∗ (1−x)^n / (1−(1−x∗y)∗w)^(n+1))
proof −
  have cbox: cbox (0, 0) (1, 1) = ({0..1} × {0..1} :: (real × real) set)
    by (auto simp: cbox-def Basis-prod-def inner-prod-def)
  have box: box (0, 0) (1, 1) = ({0<..<1} × {0<..<1} :: (real × real) set)
    by (auto simp: box-def Basis-prod-def inner-prod-def)

  have set-integrable lborel
         (cbox (0,0) (1,1)) (λ(x,y). P x ∗ (1−y)^n/(1−(1−x∗y)∗w))
    unfolding lborel-prod case-prod-unfold P-def
  proof (intro continuous-on-imp-set-integrable-cbox continuous-intros ballI)
    fix p :: real × real assume p: p ∈ cbox (0,0) (1,1)
    have (1 − fst p ∗ snd p) ∗ w ≤ 1 ∗ w
      using p cbox w by (intro mult-right-mono) auto
    also have ... < 1 using w by simp
    finally have (1 − fst p ∗ snd p) ∗ w < 1 by simp
    thus 1 − (1 − fst p ∗ snd p) ∗ w ≠ 0 by simp
  qed
  hence integrable: set-integrable lborel D (λ(x,y). P x ∗ (1−y)^n/(1−(1−x∗y)∗w))
    by (rule set-integrable-subset) (auto simp: D-def simp flip: borel-prod)

  have set-integrable lborel (cbox (0,0) (1,1))
         (λ(x,y). (1−y)^n ∗ (x∗y∗w)^n ∗ (1−x)^n / (1−(1−x∗y)∗w)^(n+1))
    unfolding lborel-prod case-prod-unfold P-def
```

```
  proof (intro continuous-on-imp-set-integrable-cbox continuous-intros ballI)
    fix p :: real × real assume p: p ∈ cbox (0,0) (1,1)
    have (1 − fst p ∗ snd p) ∗ w ≤ 1 ∗ w
      using p cbox w by (intro mult-right-mono) auto
    also have . . . < 1 using w by simp
    finally have (1 − fst p ∗ snd p) ∗ w < 1 by simp
    thus (1 − (1 − fst p ∗ snd p) ∗ w) ^ (n + 1) ≠ 0 by simp
  qed
  hence integrable′: set-integrable lborel D
               (λ(x,y). (1−y)^n ∗ (x∗y∗w)^n ∗ (1−x)^n / (1−(1−x∗y)∗w)^(n+1))
    by (rule set-integrable-subset) (auto simp: D-def simp flip: borel-prod)

  have (LBINT (x,y):D. P x ∗ (1−y)^n / (1−(1−x∗y)∗w)) =
          (LBINT y=0..1. (LBINT x=0..1. P x ∗ (1−y)^n / (1−(1−x∗y)∗w)))
    using integrable unfolding case-prod-unfold D-def lborel-prod [symmetric]
    by (subst lborel-pair.set-integral-snd) (auto simp: interval-integral-Ioo)
  also have . . . = (LBINT y=0..1. (1−y)^n ∗ (LBINT x=0..1. P x / (1−(1−y∗x)∗w)))
    by (subst interval-lebesgue-integral-mult-right [symmetric]) (auto simp: mult-ac)
  also have . . . = (LBINT y=0..1. (1−y)^n ∗ (LBINT x=0..1. (x∗y∗w)^n ∗
(1−x)^n / (1−(1−x∗y)∗w)^(n+1)))
    using w by (intro interval-lebesgue-integral-lborel-01-cong, subst beukers-aux-by-parts)
(auto simp: mult-ac)
  also have . . . = (LBINT y=0..1. (LBINT x=0..1.
                     (1−y)^n ∗ (x∗y∗w)^n ∗ (1−x)^n / (1−(1−x∗y)∗w)^(n+1)))
    by (subst interval-lebesgue-integral-mult-right [symmetric]) (auto simp: mult-ac)
  also have . . . = (LBINT (x,y):D. (1−y)^n ∗ (x∗y∗w)^n ∗ (1−x)^n / (1−(1−x∗y)∗w)^(n+1))
    using integrable′ unfolding case-prod-unfold D-def lborel-prod [symmetric]
    by (subst lborel-pair.set-integral-snd) (auto simp: interval-integral-Ioo)
  finally show ?thesis .
qed
```

We need to show the existence of some of these triple integrals.

```
lemma beukers-aux-integrable1:
  set-integrable lborel (({0<..<1} × {0<..<1}) × {0<..<1})
    (λ((x,y),z). P x ∗ P y / (1−(1−x∗y)∗z))
proof −
  have D [measurable]: D ∈ sets (borel ⨂M borel)
    unfolding D-def by (simp flip: borel-prod)
  have bounded (P ‘ {0..1})
     unfolding P-def by (intro compact-imp-bounded compact-continuous-image
continuous-intros) auto
  then obtain C where C: ⋀x. x ∈ {0..1} ⟹ norm (P x) ≤ C
    unfolding bounded-iff by fast
  show ?thesis unfolding D′-def case-prod-unfold
  proof (subst lborel-prod [symmetric],
        intro lborel-pair.Fubini-set-integrable AE-I2 impI; clarsimp?)
    fix x y :: real
    assume xy: x > 0 x < 1 y > 0 y < 1
    have x ∗ y < 1 using xy mult-strict-mono[of x 1 y 1] by simp
```

show *set-integrable lborel* {*0*<..<*1*} (λ*z*. *P x* ∗ *P y* / (*1*−(*1*−*x*∗*y*)∗*z*))
by (*rule set-integrable-subset*[*of* - {*0*..*1*}], *rule borel-integrable-atLeastAtMost*′)
(*use* ‹*x*∗*y*<*1*› *beukers-denom-neq*[*of x y*] *xy* **in** ‹*auto intro*!: *continuous-intros simp*: *P-def*›)
next
have *set-integrable lborel D*
(λ(*x*,*y*). (∫ *z*∈{*0*<..<*1*}. *norm* (*P x* ∗ *P y* / (*1*−(*1*−*x*∗*y*)∗*z*)) ∂*lborel*))
proof (*rule set-integrable-bound*[*OF* - - *AE-I2*]; *clarify?*)
show *set-integrable lborel D* (λ(*x*,*y*). C^2 ∗ (−*ln* (*x*∗*y*) / (*1* − *x*∗*y*)))
using *beukers-integral1-integrable*[*of 0 0*]
unfolding *case-prod-unfold* **by** (*intro set-integrable-mult-right*) (*auto simp*: *D-def*)
next
fix *x y* **assume** *xy*: (*x*, *y*) ∈ *D*
have *norm* (*LBINT z*:{*0*<..<*1*}. *norm* (*P x* ∗ *P y* / (*1*−(*1*−*x*∗*y*)∗*z*))) =
norm (*LBINT z*:{*0*<..<*1*}. |*P x*| ∗ |*P y*| ∗ (*1* / (*1*−(*1*−*x*∗*y*)∗*z*)))
proof (*intro arg-cong*[**where** *f* = *norm*] *set-lebesgue-integral-cong allI impI*, *goal-cases*)
case (*2 z*)
with *beukers-denom-ineq*[*of x y z*] *xy* **show** *?case*
by (*auto simp*: *abs-mult D-def*)
qed (*auto simp*: *abs-mult D-def*)
also have . . . = *norm* (|*P x*| ∗ |*P y*| ∗ (*LBINT z*=*0*..*1*. (*1* / (*1*−(*1*−*x*∗*y*)∗*z*))))
by (*subst set-integral-mult-right*) (*auto simp*: *interval-integral-Ioo*)
also have . . . = *norm* (*norm* (*P x*) ∗ *norm* (*P y*) ∗ (− *ln* (*x* ∗ *y*) / (*1* − *x* ∗ *y*)))
using *beukers-aux-ln-conv-integral*[*of x y*] *xy* **by** (*simp add*: *D-def*)
also have . . . = *norm* (*P x*) ∗ *norm* (*P y*) ∗ (− *ln* (*x* ∗ *y*) / (*1* − *x* ∗ *y*))
using *xy mult-strict-mono*[*of x 1 y 1*]
by (*auto simp*: *D-def divide-nonpos-nonneg abs-mult*)
also have *norm* (*P x*) ∗ *norm* (*P y*) ∗ (− *ln* (*x* ∗ *y*) / (*1* − *x* ∗ *y*)) ≤
norm (*C* ∗ *C* ∗ (− *ln* (*x* ∗ *y*) / (*1* − *x* ∗ *y*)))
using *xy C*[*of x*] *C*[*of y*] *mult-strict-mono*[*of x 1 y 1*] **unfolding** *norm-mult norm-divide*
by (*intro mult-mono C*) (*auto simp*: *D-def divide-nonpos-nonneg*)
finally show *norm* (*LBINT z*:{*0*<..<*1*}. *norm* (*P x* ∗ *P y* / (*1*−(*1*−*x*∗*y*)∗*z*)))
≤ *norm* (*case* (*x*, *y*) *of* (*x*, *y*) ⇒ C^2 ∗ (− *ln* (*x* ∗ *y*) / (*1* − *x* ∗ *y*)))
by (*simp add*: *power2-eq-square mult-ac*)
next
show *set-borel-measurable lborel D* (λ(*x*, *y*).
LBINT z:{*0*<..<*1*}. *norm* (*P x* ∗ *P y* / (*1* − (*1* − *x* ∗ *y*) ∗ *z*)))
unfolding *lborel-prod* [*symmetric*] *set-borel-measurable-def*
case-prod-unfold set-lebesgue-integral-def P-def
by *measurable*
qed
thus *set-integrable lborel* ({*0*<..<*1*} × {*0*<..<*1*})
(λ*x*. *LBINT y*:{*0*<..<*1*}. |*P* (*fst x*) ∗ *P* (*snd x*)| / |*1* − (*1* − *fst x* ∗ *snd x*) ∗ *y*|)
by (*simp add*: *case-prod-unfold D-def*)

qed (auto simp: case-prod-unfold lborel-prod [symmetric] set-borel-measurable-def P-def)
qed

lemma beukers-aux-integrable2:
set-integrable lborel D′ (λ(z,x,y). P x * (x*y*z) ^ n * (1−y) ^ n / (1−(1−x*y)*z) ^ (n+1))
proof −
have [measurable]: P ∈ borel-measurable borel **unfolding** P-def
by (intro borel-measurable-continuous-onI continuous-intros)
have bounded (P ‘ {0..1})
unfolding P-def **by** (intro compact-imp-bounded compact-continuous-image continuous-intros) auto
then obtain C **where** C: ⋀x. x ∈ {0..1} ⟹ norm (P x) ≤ C
unfolding bounded-iff **by** fast
show ?thesis **unfolding** D′-def
proof (rule set-integrable-bound[OF - - AE-I2]; clarify?)
show set-integrable lborel ({0<..<1} × {0<..<1} × {0<..<1})
(λ(z,x,y). C * (1 / (1−(1−x*y)*z)))
unfolding case-prod-unfold
using beukers-integral1-integrable′[of 0 0]
by (intro set-integrable-mult-right) (auto simp: lborel-prod case-prod-unfold)
next
fix x y z :: real **assume** xyz: x ∈ {0<..<1} y ∈ {0<..<1} z ∈ {0<..<1}
have norm (P x * (x*y*z) ^ n * (1−y) ^ n / (1−(1−x*y)*z) ^ (n+1)) =
norm (P x) * (1−y) ^ n * ((x*y*z) / (1−(1−x*y)*z)) ^ n / (1−(1−x*y)*z)
using xyz beukers-denom-ineq[of x y z] **by** (simp add: abs-mult power-divide mult-ac)
also have (x*y*z) / (1−(1−x*y)*z) = 1/((1−z)/(z*x*y)+1)
using xyz **by** (simp add: field-simps)
also have norm (P x) * (1−y) ^ n * ... ^ n / (1−(1−x*y)*z) ≤
C * 1 ^ n * 1 ^ n / (1−(1−x*y)*z)
using xyz C[of x] beukers-denom-ineq[of x y z]
by (intro mult-mono divide-right-mono power-mono zero-le-power mult-nonneg-nonneg divide-nonneg-nonneg)
(auto simp: field-simps)
also have ... ≤ |C * 1 ^ n * 1 ^ n / (1−(1−x*y)*z)|
by linarith
finally show norm (P x * (x*y*z) ^ n * (1−y) ^ n / (1−(1−x*y)*z) ^ (n+1)) ≤
norm (case (z,x,y) of (z,x,y) ⇒ C * (1 / (1−(1−x*y)*z)))
by (simp add: case-prod-unfold)
qed (simp-all add: case-prod-unfold set-borel-measurable-def flip: borel-prod)
qed

lemma beukers-aux-integrable3:
set-integrable lborel D′ (λ(w,x,y). P x * (1−w) ^ n * (1−y) ^ n / (1−(1−x*y)*w))
proof −
have [measurable]: P ∈ borel-measurable borel **unfolding** P-def
by (intro borel-measurable-continuous-onI continuous-intros)

have bounded (P ‘ {0..1})
unfolding P-def **by** (intro compact-imp-bounded compact-continuous-image continuous-intros) auto
then obtain C **where** C: ⋀x. x ∈ {0..1} ⟹ norm (P x) ≤ C
unfolding bounded-iff **by** fast
show ?thesis **unfolding** D′-def
proof (rule set-integrable-bound[OF - - AE-I2]; clarify?)
show set-integrable lborel ({0<..<1} × {0<..<1} × {0<..<1})
(λ(z,x,y). C ∗ (1 / (1−(1−x∗y)∗z)))
unfolding case-prod-unfold
using beukers-integral1-integrable′[of 0 0]
by (intro set-integrable-mult-right) (auto simp: lborel-prod case-prod-unfold)
next
fix x y w :: real **assume** xyw: x ∈ {0<..<1} y ∈ {0<..<1} w ∈ {0<..<1}
have norm (P x ∗ (1−w)^n ∗ (1−y)^n / (1−(1−x∗y)∗w)) =
norm (P x) ∗ (1−w)^n ∗ (1−y)^n / (1−(1−x∗y)∗w)
using xyw beukers-denom-ineq[of x y w] **by** (simp add: abs-mult power-divide mult-ac)
also have ... ≤ C ∗ 1^n ∗ 1^n / (1−(1−x∗y)∗w)
using xyw C[of x] beukers-denom-ineq[of x y w]
by (intro mult-mono divide-right-mono power-mono zero-le-power mult-nonneg-nonneg divide-nonneg-nonneg)
(auto simp: field-simps)
also have ... ≤ |C ∗ 1^n ∗ 1^n / (1−(1−x∗y)∗w)|
by linarith
finally show norm (P x ∗ (1−w)^n ∗ (1−y)^n / (1−(1−x∗y)∗w)) ≤
norm (case (w,x,y) of (z,x,y) ⇒ C ∗ (1 / (1−(1−x∗y)∗z)))
by (simp add: case-prod-unfold)
qed (simp-all add: case-prod-unfold set-borel-measurable-def flip: borel-prod)
qed

Now we only need to put all of these results together:

lemma beukers-integral2-conv-3: beukers-integral2 = beukers-integral3
proof −
have cont-P: continuous-on {0..1} P
by (auto simp: P-def intro: continuous-intros)
have D [measurable]: D ∈ sets borel D ∈ sets (borel ⨂ $_M$ borel)
unfolding D-def **by** (simp-all flip: borel-prod)
have [measurable]: P ∈ borel-measurable borel **unfolding** P-def
by (intro borel-measurable-continuous-onI continuous-intros)

have beukers-integral2 = (LBINT (x,y):D. P x ∗ P y ∗ (LBINT z=0..1. 1 / (1−(1−x∗y)∗z)))
unfolding beukers-integral2-def case-prod-unfold
by (intro set-lebesgue-integral-cong allI impI, measurable)
(subst beukers-aux-ln-conv-integral, auto simp: D-def)
also have ... = (LBINT (x,y):D. (LBINT z=0..1. P x ∗ P y / (1−(1−x∗y)∗z)))
by (subst interval-lebesgue-integral-mult-right [symmetric]) auto
also have ... = (LBINT (x,y):D. (LBINT z:{0<..<1}. P x ∗ P y / (1−(1−x∗y)∗z)))

```
    by (simp add: interval-integral-Ioo)
  also have ... = (LBINT z:{0<..<1}. (LBINT (x,y):D. P x ∗ P y / (1−(1−x∗y)∗z)))
   proof (subst lborel-pair.Fubini-set-integral [symmetric])
    have set-integrable lborel (({0<..<1} × {0<..<1}) × {0<..<1})
          (λ((x, y), z). P x ∗ P y / (1 − (1 − x ∗ y) ∗ z))
      using beukers-aux-integrable1 by simp
    also have ?this ⟷ set-integrable (lborel ⨂M lborel) ({0<..<1} × D)
                        (λ(z,x,y). P x ∗ P y / (1 − (1 − x ∗ y) ∗ z))
      unfolding set-integrable-def
     by (subst lborel-pair.integrable-product-swap-iff [symmetric], intro Bochner-Integration.integrable-cong)
        (auto simp: indicator-def case-prod-unfold lborel-prod D-def)
    finally show ... .
  qed (auto simp: case-prod-unfold)
   also have ... = (LBINT z:{0<..<1}. (LBINT (x,y):D. P x ∗ (x∗y∗z)^n ∗
(1−y)^n / (1−(1−x∗y)∗z)^(n+1)))
      by (rule set-lebesgue-integral-cong) (use beukers-aux-integral-transform1 in
simp-all)
   also have ... = (LBINT (x,y):D. (LBINT z:{0<..<1}. P x ∗ (x∗y∗z)^n ∗
(1−y)^n / (1−(1−x∗y)∗z)^(n+1)))
    using beukers-aux-integrable2
    by (subst lborel-pair.Fubini-set-integral [symmetric])
       (auto simp: case-prod-unfold lborel-prod D-def D′-def)
   also have ... = (LBINT (x,y):D. (LBINT w:{0<..<1}. P x ∗ (1−w)^n ∗
(1−y)^n / (1−(1−x∗y)∗w)))
  proof (intro set-lebesgue-integral-cong allI impI; clarify?)
    fix x y :: real
    assume xy: (x, y) ∈ D
   have (LBINT z:{0<..<1}. P x ∗ (x∗y∗z)^n ∗ (1−y)^n / (1−(1−x∗y)∗z)^(n+1))
=
          P x ∗ (LBINT z=0..1. (x∗y∗z)^n ∗ (1−y)^n / (1−(1−x∗y)∗z)^(n+1))
      by (subst interval-lebesgue-integral-mult-right [symmetric])
         (simp add: mult-ac interval-integral-Ioo)
   also have ... = P x ∗ (LBINT w=0..1. (1−w)^n ∗ (1−y)^n / (1−(1−x∗y)∗w))
      using xy by (subst beukers-aux-integral-transform2) (auto simp: D-def)
   also have ... = (LBINT w:{0<..<1}. P x ∗ (1−w)^n ∗ (1−y)^n / (1−(1−x∗y)∗w))
      by (subst interval-lebesgue-integral-mult-right [symmetric])
         (simp add: mult-ac interval-integral-Ioo)
   finally show (LBINT z:{0<..<1}. P x ∗ (x∗y∗z)^n ∗ (1−y)^n / (1−(1−x∗y)∗z)^(n+1))
=
             (LBINT w:{0<..<1}. P x ∗ (1−w)^n ∗ (1−y)^n / (1−(1−x∗y)∗w))
.
  qed (auto simp: D-def simp flip: borel-prod)
   also have ... = (LBINT w:{0<..<1}. (LBINT (x,y):D. P x ∗ (1−w)^n ∗
(1−y)^n / (1−(1−x∗y)∗w)))
    using beukers-aux-integrable3
    by (subst lborel-pair.Fubini-set-integral [symmetric])
       (auto simp: case-prod-unfold lborel-prod D-def D′-def)
  also have ... = (LBINT w=0..1. (1−w)^n ∗ (LBINT (x,y):D. P x ∗ (1−y)^n
/ (1−(1−x∗y)∗w)))
```

unfolding *case-prod-unfold*
by (*subst set-integral-mult-right* [*symmetric*]) (*simp add*: *mult-ac interval-integral-Ioo*)
also have ... = (*LBINT w=0..1*. (*1−w*) ^ *n* ∗ (*LBINT* (*x*,*y*):*D*. (*x*∗*y*∗*w*∗(*1−x*)∗(*1−y*)) ^ *n* / (*1*−(*1−x*∗*y*)∗*w*) ^ (*n+1*)))
by (*intro interval-lebesgue-integral-lborel-01-cong*, *subst beukers-aux-integral-transform3*)
(*auto simp*: *mult-ac power-mult-distrib*)
also have ... = (*LBINT w=0..1*. (*LBINT* (*x*,*y*):*D*. (*x*∗*y*∗*w*∗(*1−x*)∗(*1−y*)∗(*1−w*)) ^ *n* / (*1*−(*1−x*∗*y*)∗*w*) ^ (*n+1*)))
by (*subst set-integral-mult-right* [*symmetric*])
(*auto simp*: *case-prod-unfold mult-ac power-mult-distrib*)
also have ... = *beukers-integral3*
using *beukers-integral3-integrable* **unfolding** *D′-def D-def beukers-integral3-def*
by (*subst* (*2*) *lborel-prod* [*symmetric*], *subst lborel-pair.set-integral-fst′*)
(*auto simp*: *case-prod-unfold interval-integral-Ioo lborel-prod algebra-simps*)
finally show *?thesis* **.**
qed

1.10 The main result

Combining all of the results so far, we can derive the key inequalities

$$0 < A\zeta(3) + B < 2\zeta(3) \cdot 27^{-n} \cdot \mathrm{lcm}\{1 \dots n\}^3$$

for integers A, B with $A > 0$.

lemma *zeta-3-linear-combination-bounds*:
obtains *A B* :: *int*
where *A* > *0*
A ∗ *Re* (*zeta 3*) + *B* ∈ {*0* <.. *2* ∗ *Re* (*zeta 3*) ∗ *Lcm* {*1..n*} ^ *3* / *27* ^ *n*}
proof −
define *I* **where** *I* = *beukers-integral2*
define *d* **where** *d* = *Lcm* {*1..n*} ^ *3*
have *d* > *0* **by** (*auto simp*: *d-def intro*!: *Nat.gr0I*)
from *beukers-integral2-conv-int-combination* **obtain** *A′ B* :: *int*
where ∗: *A′* > *0 I* = *A′* ∗ *Re* (*zeta 3*) + *B* / *d* **unfolding** *I-def d-def* **.**
define *A* **where** *A* = *A′* ∗ *d*
from ∗ **have** *A*: *A* > *0 I* = (*A* ∗ *Re* (*zeta 3*) + *B*) / *d*
using ‹*d* > *0*› **by** (*simp-all add*: *A-def field-simps*)

have *0* < *I*
using *beukers-integral3-pos* **by** (*simp add*: *I-def beukers-integral2-conv-3*)
with ‹*d* > *0*› **have** *A* ∗ *Re* (*zeta 3*) + *B* > *0*
by (*simp add*: *field-simps A*)

moreover have *I* ≤ *2* ∗ (*1* / *27*) ^ *n* ∗ *Re* (*zeta 3*)
using *beukers-integral2-conv-3 beukers-integral3-le* **by** (*simp add*: *I-def*)
hence *A* ∗ *Re* (*zeta 3*) + *B* ≤ *2* ∗ *Re* (*zeta 3*) ∗ *d* / *27* ^ *n*
using ‹*d* > *0*› **by** (*simp add*: *A field-simps*)

ultimately show *?thesis*

using *A* **by** (*intro that*[*of A B*]) (*auto simp*: *d-def*)
qed

If $\zeta(3) = \frac{a}{b}$ for some integers a and b, we can thus derive the inequality $2b\zeta(3) \cdot 27^{-n} \cdot \mathrm{lcm}\{1 \ldots n\}^3 \geq 1$ for any natural number n.

lemma *beukers-key-inequality*:
fixes *a* :: *int* **and** *b* :: *nat*
assumes *b* > *0* **and** *ab*: *Re* (*zeta 3*) = *a* / *b*
shows *2* ∗ *b* ∗ *Re* (*zeta 3*) ∗ *Lcm* {*1..n*} ^ *3* / *27* ^ *n* ≥ *1*
proof −
from *zeta-3-linear-combination-bounds* **obtain** *A B* :: *int*
where *AB*: *A* > *0*
A ∗ *Re* (*zeta 3*) + *B* ∈ {*0* <.. *2* ∗ *Re* (*zeta 3*) ∗ *Lcm* {*1..n*} ^ *3* / *27* ^ *n*} **.**
from *AB* **have** *0* < (*A* ∗ *Re* (*zeta 3*) + *B*) ∗ *b*
using ‹*b* > *0*› **by** (*intro mult-pos-pos*) *auto*
also have ... = *A* ∗ (*Re* (*zeta 3*) ∗ *b*) + *B* ∗ *b*
using ‹*b* > *0*› **by** (*simp add*: *algebra-simps*)
also have *Re* (*zeta 3*) ∗ *b* = *a*
using ‹*b* > *0*› **by** (*simp add*: *ab*)
also have *of-int A* ∗ *of-int a* + *of-int* (*B* ∗ *b*) = *of-int* (*A* ∗ *a* + *B* ∗ *b*)
by *simp*
finally have *1* ≤ *A* ∗ *a* + *B* ∗ *b*
by *linarith*
hence *1* ≤ *real-of-int* (*A* ∗ *a* + *B* ∗ *b*)
by *linarith*
also have ... = (*A* ∗ (*a* / *b*) + *B*) ∗ *b*
using ‹*b* > *0*› **by** (*simp add*: *ring-distribs*)
also have *a* / *b* = *Re* (*zeta 3*)
by (*simp add*: *ab*)
also have *A* ∗ *Re* (*zeta 3*) + *B* ≤ *2* ∗ *Re* (*zeta 3*) ∗ *Lcm* {*1..n*} ^ *3* / *27* ^ *n*
using *AB* **by** *simp*
finally show *2* ∗ *b* ∗ *Re* (*zeta 3*) ∗ *Lcm* {*1..n*} ^ *3* / *27* ^ *n* ≥ *1*
using ‹*b* > *0*› **by** (*simp add*: *mult-ac*)
qed

end

lemma *smallo-power*: *n* > *0* ⟹ *f* ∈ *o*[*F*](*g*) ⟹ (λ*x*. *f x* ^ *n*) ∈ *o*[*F*](λ*x*. *g x* ^ *n*)
by (*induction n rule*: *nat-induct-non-zero*) (*auto intro*: *landau-o.small.mult*)

This is now a contradiction, since $\mathrm{lcm}\{1 \ldots n\} \in o(3^n)$ by the Prime Number Theorem – hence the main result.

theorem *zeta-3-irrational*: *zeta 3* ∉ ℚ
proof
assume *zeta 3* ∈ ℚ
obtain *a* :: *int* **and** *b* :: *nat* **where** *b* > *0* **and** *ab'*: *zeta 3* = *a* / *b*

proof −
from ‹*zeta 3* ∈ ℚ› **obtain** *r* **where** *r*: *zeta 3* = *of-rat r*
by (*elim Rats-cases*)
also have *r* = *rat-of-int* (*fst* (*quotient-of r*)) / *rat-of-int* (*snd* (*quotient-of r*))
by (*intro quotient-of-div*) *auto*
also have *of-rat* ... = (*of-int* (*fst* (*quotient-of r*)) / *of-int* (*snd* (*quotient-of r*)) :: *complex*)
by (*simp add*: *of-rat-divide*)
also have *of-int* (*snd* (*quotient-of r*)) = *of-nat* (*nat* (*snd* (*quotient-of r*)))
using *quotient-of-denom-pos'*[*of r*] **by** *auto*
finally have *zeta 3* = *of-int* (*fst* (*quotient-of r*)) / *of-nat* (*nat* (*snd* (*quotient-of r*))) **.**
thus *?thesis*
using *quotient-of-denom-pos'*[*of r*]
by (*intro that*[*of nat* (*snd* (*quotient-of r*)) *fst* (*quotient-of r*)]) *auto*
qed
hence *ab*: *Re* (*zeta 3*) = *a* / *b* **by** *simp*

interpret *prime-number-theorem*
by *standard* (*rule prime-number-theorem*)

have *Lcm-upto-smallo*: (λ*n*. *real* (*Lcm* {*1..n*})) ∈ *o*(λ*n*. *c* ^ *n*) **if** *c*: *c* > *exp 1* **for** *c*
proof −
have *0* < *exp* (*1*::*real*) **by** *simp*
also note *c*
finally have *c* > *0* **.**
have (λ*n*. *real* (*Lcm* {*1..n*})) = (λ*n*. *real* (*Lcm* {*1..nat* ⌊*real n*⌋}))
by *simp*
also have ... ∈ *o*(λ*n*. *c powr real n*)
using *Lcm-upto.smallo'*
by (*rule landau-o.small.compose*) (*simp-all add*: *c filterlim-real-sequentially*)
also have (λ*n*. *c powr real n*) = (λ*n*. *c* ^ *n*)
using *c* ‹*c* > *0*› **by** (*subst powr-realpow*) *auto*
finally show *?thesis* **.**
qed

have (λ*n*. *2* ∗ *b* ∗ *Re* (*zeta 3*) ∗ *real* (*Lcm* {*1..n*}) ^ *3* / *27* ^ *n*) ∈
O(λ*n*. *real* (*Lcm* {*1..n*}) ^ *3* / *27* ^ *n*)
using ‹*b* > *0*› *Re-zeta-ge-1*[*of 3*] **by** *simp*
also have *exp 1* < (*3* :: *real*)
using *e-approx-32* **by** (*simp add*: *abs-if split*: *if-splits*)
hence (λ*n*. *real* (*Lcm* {*1..n*}) ^ *3* / *27* ^ *n*) ∈ *o*(λ*n*. (*3* ^ *n*) ^ *3* / *27* ^ *n*)
unfolding *of-nat-power*
by (*intro landau-o.small.divide-right smallo-power Lcm-upto-smallo*) *auto*
also have (λ*n*. (*3* ^ *n*) ^ *3* / *27* ^ *n* :: *real*) = (λ-. *1*)
by (*simp add*: *power-mult* [*of 3*, *symmetric*] *mult.commute*[*of - 3*] *power-mult*[*of - 3*])
finally have ∗: (λ*n*. *2* ∗ *b* ∗ *Re* (*zeta 3*) ∗ *real* (*Lcm* {*1..n*}) ^ *3* / *27* ^ *n*) ∈

$o(\lambda\text{-}.\ 1)$.

have *lim*: $(\lambda n.\ 2 * b * Re\ (zeta\ 3) * real\ (Lcm\ \{1..n\})\ \hat{}\ 3\ /\ 27\ \hat{}\ n) \longrightarrow 0$
using *smalloD-tendsto*[*OF* ∗] **by** *simp*

moreover have $1 \leq real\ (2 * b) * Re\ (zeta\ 3) * real\ (Lcm\ \{1..n\}\ \hat{}\ 3)\ /\ 27\ \hat{}\ n$ **for** *n*
using *beukers-key-inequality*[*of b a*] *ab* ‹*b* > *0*› **by** *simp*

ultimately have $1 \leq (0 :: real)$
by (*intro tendsto-lowerbound*[*OF lim*] *always-eventually allI*) *auto*
thus *False* **by** *simp*
qed

end

References

[1] R. Apéry. Irrationalité de $\zeta 2$ et $\zeta 3$. In *Journées Arithmétiques de Luminy*, number 61 in Astérisque, pages 11–13. Société mathématique de France, 1979.

[2] F. Beukers. A note on the irrationality of $\zeta(2)$ and $\zeta(3)$. *Bulletin of the London Mathematical Society*, 11(3):268–272, 1979.

[3] M. Eberl. Elementary facts about the distribution of primes. *Archive of Formal Proofs*, Feb. 2019. http://isa-afp.org/entries/Prime_Distribution_Elementary.html, Formal proof development.

[4] M. Eberl and L. C. Paulson. The prime number theorem. *Archive of Formal Proofs*, Sept. 2018. http://isa-afp.org/entries/Prime_Number_Theorem.html, Formal proof development.

[5] M. Filaseta. Math 785: Transcendental number theory (lecture notes, part 4), 2011.

[6] A. Mahboubi and T. Sibut-Pinote. A formal proof of the irrationality of $\zeta(3)$, 2019.