The Prime Number Theorem

Manuel Eberl and Larry Paulson

March 17, 2025

Abstract

This article provides a short proof of the Prime Number Theorem in several equivalent forms, most notably $\pi(x) \sim x/\ln x$ where $\pi(x)$ is the number of primes no larger than x. It also defines other basic number-theoretic functions related to primes like Chebyshev's ϑ and ψ and the "n-th prime number" function p_n. We also show various bounds and relationship between these functions are shown. Lastly, we derive Mertens' First and Second Theorem, i. e. $\sum_{p\leq x} \frac{\ln p}{p} = \ln x + O(1)$ and $\sum_{p\leq x} \frac{1}{p} = \ln\ln x + M + O(1/\ln x)$. We also give explicit bounds for the remainder terms.

The proof of the Prime Number Theorem builds on a library of Dirichlet series and analytic combinatorics. We essentially follow the presentation by Newman [6]. The core part of the proof is a Tauberian theorem for Dirichlet series, which is proven using complex analysis and then used to strengthen Mertens' First Theorem to $\sum_{p\leq x} \frac{\ln p}{p} = \ln x + c + o(1)$.

A variant of this proof has been formalised before by Harrison in HOL Light [5], and formalisations of Selberg's elementary proof exist both by Avigad *et al.* [2] in Isabelle and by Carneiro [3] in Metamath. The advantage of the analytic proof is that, while it requires more powerful mathematical tools, it is considerably shorter and clearer. This article attempts to provide a short and clear formalisation of all components of that proof using the full range of mathematical machinery available in Isabelle, staying as close as possible to Newman's simple paper proof.

Contents

1 Auxiliary material

theory *Prime-Number-Theorem-Library*
imports
Zeta-Function.Zeta-Function
HOL−Real-Asymp.Real-Asymp
begin

Conflicting notation from *HOL−Analysis.Infinite-Sum*

no-notation *Infinite-Sum.abs-summable-on* (**infixr** ‹*abs′-summable′-on*› *46*)

lemma *homotopic-loopsI*:
fixes $h :: real \times real \Rightarrow -$
assumes *continuous-on* $(\{0..1\} \times \{0..1\})$ h
h ‘ $(\{0..1\} \times \{0..1\}) \subseteq s$
$\bigwedge x.\ x \in \{0..1\} \Longrightarrow h\ (0,\ x) = p\ x$
$\bigwedge x.\ x \in \{0..1\} \Longrightarrow h\ (1,\ x) = q\ x$
$\bigwedge x.\ x \in \{0..1\} \Longrightarrow$ *pathfinish* $(h \circ Pair\ x) =$ *pathstart* $(h \circ Pair\ x)$
shows *homotopic-loops s p q*
using *assms* **unfolding** *homotopic-loops* **by** (*intro exI*[*of* - *h*]) *auto*

lemma *homotopic-pathsI*:
fixes $h :: real \times real \Rightarrow -$
assumes *continuous-on* $(\{0..1\} \times \{0..1\})$ h
assumes h ‘ $(\{0..1\} \times \{0..1\}) \subseteq s$
assumes $\bigwedge x.\ x \in \{0..1\} \Longrightarrow h\ (0,\ x) = p\ x$
assumes $\bigwedge x.\ x \in \{0..1\} \Longrightarrow h\ (1,\ x) = q\ x$
assumes $\bigwedge x.\ x \in \{0..1\} \Longrightarrow$ *pathstart* $(h \circ Pair\ x) =$ *pathstart* p
assumes $\bigwedge x.\ x \in \{0..1\} \Longrightarrow$ *pathfinish* $(h \circ Pair\ x) =$ *pathfinish* p
shows *homotopic-paths s p q*
using *assms* **unfolding** *homotopic-paths* **by** (*intro exI*[*of* - *h*]) *auto*

lemma *sum-upto-ln-conv-sum-upto-mangoldt*:
sum-upto $(\lambda n.\ ln\ (real\ n))\ x =$ *sum-upto* $(\lambda n.\ mangoldt\ n * nat\ \lfloor x\ /\ real\ n \rfloor)\ x$
proof −
have *sum-upto* $(\lambda n.\ ln\ (real\ n))\ x =$
sum-upto $(\lambda n.\ \sum d \mid d\ dvd\ n.\ mangoldt\ d)\ x$
by (*intro sum-upto-cong*) (*simp-all add*: *mangoldt-sum*)
also have $\ldots =$ *sum-upto* $(\lambda k.\ sum\text{-}upto\ (\lambda d.\ mangoldt\ k)\ (x\ /\ real\ k))\ x$
by (*rule sum-upto-sum-divisors*)
also have $\ldots =$ *sum-upto* $(\lambda n.\ mangoldt\ n * nat\ \lfloor x\ /\ real\ n \rfloor)\ x$
unfolding *sum-upto-altdef* **by** (*simp add*: *mult-ac*)
finally show *?thesis* **.**
qed

lemma *ln-fact-conv-sum-upto-mangoldt*:
$ln\ (fact\ n) = sum\text{-}upto\ (\lambda k.\ mangoldt\ k * (n\ div\ k))\ n$
proof −
have [*simp*]: $\{0<..Suc\ n\} = insert\ (Suc\ n)\ \{0<..n\}$ **for** n **by** *auto*

```
  have ln (fact n) = sum-upto (λn. ln (real n)) n
    by (induction n) (auto simp: sum-upto-altdef nat-add-distrib ln-mult)
  also have ... = sum-upto (λk. mangoldt k * (n div k)) n
    unfolding sum-upto-ln-conv-sum-upto-mangoldt
    by (intro sum-upto-cong) (auto simp: floor-divide-of-nat-eq)
  finally show ?thesis .
qed

lemma fds-abs-converges-comparison-test:
  fixes s :: 'a :: dirichlet-series
  assumes eventually (λn. norm (fds-nth f n) ≤ fds-nth g n) at-top and fds-converges
g (s - 1)
  shows   fds-abs-converges f s
  unfolding fds-abs-converges-def
proof (rule summable-comparison-test-ev)
  from assms(2) show summable (λn. fds-nth g n / n powr (s - 1))
    by (auto simp: fds-converges-def)
  from assms(1) eventually-gt-at-top[of 0]
    show eventually (λn. norm (norm (fds-nth f n / nat-power n s)) ≤
                          fds-nth g n / real n powr (s - 1)) at-top
   by eventually-elim (auto simp: norm-divide norm-nat-power intro!: divide-right-mono)
qed

lemma fds-converges-scaleR [intro]:
  assumes fds-converges f s
  shows   fds-converges (c *R f) s
proof -
  from assms have summable (λn. c *R (fds-nth f n / nat-power n s))
    by (intro summable-scaleR-right) (auto simp: fds-converges-def)
  also have (λn. c *R (fds-nth f n / nat-power n s)) = (λn. (c *R fds-nth f n /
nat-power n s))
    by (simp add: scaleR-conv-of-real)
  finally show ?thesis by (simp add: fds-converges-def)
qed

lemma fds-abs-converges-scaleR [intro]:
  assumes fds-abs-converges f s
  shows   fds-abs-converges (c *R f) s
proof -
  from assms have summable (λn. abs c * norm (fds-nth f n / nat-power n s))
    by (intro summable-mult) (auto simp: fds-abs-converges-def)
  also have (λn. abs c * norm (fds-nth f n / nat-power n s)) =
                (λn. norm ((c *R fds-nth f n) / nat-power n s)) by (simp add:
norm-divide)
  finally show ?thesis by (simp add: fds-abs-converges-def)
qed

lemma conv-abscissa-scaleR: conv-abscissa (scaleR c f) ≤ conv-abscissa f
  by (rule conv-abscissa-mono) auto
```

lemma *abs-conv-abscissa-scaleR*: *abs-conv-abscissa* (*scaleR c f*) ≤ *abs-conv-abscissa f*
by (*rule abs-conv-abscissa-mono*) *auto*

lemma *fds-abs-converges-mult-const-left* [*intro*]:
fds-abs-converges f s ⟹ *fds-abs-converges* (*fds-const c* ∗ *f*) *s*
by (*auto simp*: *fds-abs-converges-def norm-mult norm-divide dest*: *summable-mult*[*of - norm c*])

lemma *conv-abscissa-mult-const-left*:
conv-abscissa (*fds-const c* ∗ *f*) ≤ *conv-abscissa f*
by (*intro conv-abscissa-mono*) *auto*

lemma *abs-conv-abscissa-mult-const-left*:
abs-conv-abscissa (*fds-const c* ∗ *f*) ≤ *abs-conv-abscissa f*
by (*intro abs-conv-abscissa-mono*) *auto*

lemma *fds-abs-converges-mult-const-right* [*intro*]:
fds-abs-converges f s ⟹ *fds-abs-converges* (*f* ∗ *fds-const c*) *s*
by (*metis mult.commute fds-abs-converges-mult-const-left*)

lemma *conv-abscissa-mult-const-right*:
conv-abscissa (*f* ∗ *fds-const c*) ≤ *conv-abscissa f*
by (*intro conv-abscissa-mono*) *auto*

lemma *abs-conv-abscissa-mult-const-right*:
abs-conv-abscissa (*f* ∗ *fds-const c*) ≤ *abs-conv-abscissa f*
by (*intro abs-conv-abscissa-mono*) *auto*

lemma *bounded-coeffs-imp-fds-abs-converges*:
fixes *s* :: *'a* :: *dirichlet-series* **and** *f* :: *'a fds*
assumes *Bseq* (*fds-nth f*) *s* · *1* > *1*
shows *fds-abs-converges f s*
proof −
from *assms* **obtain** *C* **where** *C*: ⋀*n*. *norm* (*fds-nth f n*) ≤ *C*
by (*auto simp*: *Bseq-def*)
show *?thesis*
proof (*rule fds-abs-converges-comparison-test*)
from ‹*s* · *1* > *1*› **show** *fds-converges* (*C* ∗*R fds-zeta*) (*s* · *1*)
by (*intro fds-abs-converges-imp-converges*) *auto*
from *C* **show** *eventually* (λ*n*. *norm* (*fds-nth f n*) ≤ *fds-nth* (*C* ∗*R fds-zeta*) *n*) *at-top*
by (*intro always-eventually*) (*auto simp*: *fds-nth-zeta*)
qed
qed

lemma *bounded-coeffs-imp-fds-abs-converges′*:

fixes $s :: 'a :: dirichlet\text{-}series$ **and** $f :: 'a\ fds$
assumes $Bseq\ (\lambda n.\ fds\text{-}nth\ f\ n * nat\text{-}power\ n\ s0)$ $s \cdot 1 > 1 - s0 \cdot 1$
shows $fds\text{-}abs\text{-}converges\ f\ s$
proof −
have $fds\text{-}nth\ (fds\text{-}shift\ s0\ f) = (\lambda n.\ fds\text{-}nth\ f\ n * nat\text{-}power\ n\ s0)$
by (*auto simp*: *fun-eq-iff*)
with *assms* **have** $Bseq\ (fds\text{-}nth\ (fds\text{-}shift\ s0\ f))$ **by** *simp*
with *assms*(*2*) **have** $fds\text{-}abs\text{-}converges\ (fds\text{-}shift\ s0\ f)\ (s + s0)$
by (*intro bounded-coeffs-imp-fds-abs-converges*) (*auto simp*: *algebra-simps*)
thus *?thesis* **by** *simp*
qed

lemma *bounded-coeffs-imp-abs-conv-abscissa-le*:
fixes $s :: 'a :: dirichlet\text{-}series$ **and** $f :: 'a\ fds$ **and** $c :: ereal$
assumes $Bseq\ (\lambda n.\ fds\text{-}nth\ f\ n * nat\text{-}power\ n\ s)$ $1 - s \cdot 1 \leq c$
shows $abs\text{-}conv\text{-}abscissa\ f \leq c$
proof (*rule abs-conv-abscissa-leI-weak*)
fix x **assume** $c < ereal\ x$
have $ereal\ (1 - s \cdot 1) \leq c$ **by** *fact*
also have $\ldots < ereal\ x$ **by** *fact*
finally have $1 - s \cdot 1 < ereal\ x$ **by** *simp*
thus $fds\text{-}abs\text{-}converges\ f\ (of\text{-}real\ x)$
by (*intro bounded-coeffs-imp-fds-abs-converges'*[*OF assms*(*1*)]) *auto*
qed

lemma *bounded-coeffs-imp-abs-conv-abscissa-le-1*:
fixes $s :: 'a :: dirichlet\text{-}series$ **and** $f :: 'a\ fds$
assumes $Bseq\ (\lambda n.\ fds\text{-}nth\ f\ n)$
shows $abs\text{-}conv\text{-}abscissa\ f \leq 1$
proof −
have [*simp*]: $fds\text{-}nth\ f\ n * nat\text{-}power\ n\ 0 = fds\text{-}nth\ f\ n$ **for** n
by (*cases n = 0*) *auto*
show *?thesis*
by (*rule bounded-coeffs-imp-abs-conv-abscissa-le*[**where** $s = 0$]) (*insert assms*, *auto simp*:)
qed

lemma
fixes $a\ b\ c :: real$
assumes *ab*: $a + b > 0$ **and** *c*: $c < -1$
shows *set-integrable-powr-at-top*: $(\lambda x.\ (b + x)\ powr\ c)$ *absolutely-integrable-on* $\{a<..\}$
and *set-lebesgue-integral-powr-at-top*:
$(\int x \in \{a<..\}.\ ((b + x)\ powr\ c)\ \partial lborel) = -((b + a)\ powr\ (c + 1) / (c + 1))$
and *powr-has-integral-at-top*:
$((\lambda x.\ (b + x)\ powr\ c)\ has\text{-}integral\ -((b + a)\ powr\ (c + 1) / (c + 1)))$ $\{a<..\}$

proof −
let *?f* = λ*x*. (*b* + *x*) *powr c* **and** *?F* = λ*x*. (*b* + *x*) *powr* (*c* + *1*) / (*c* + *1*)
have *limits*: ((*?F* ∘ *real-of-ereal*) ⟶ *?F a*) (*at-right* (*ereal a*))
((*?F* ∘ *real-of-ereal*) ⟶ *0*) (*at-left* ∞)
using *c ab* **unfolding** *ereal-tendsto-simps1* **by** (*real-asymp simp*: *field-simps*)+
have *1*: *set-integrable lborel* (*einterval a* ∞) *?f* **using** *ab c limits*
by (*intro interval-integral-FTC-nonneg*) (*auto intro*!: *derivative-eq-intros continuous-intros*)
thus *2*: *?f absolutely-integrable-on* {*a*<..}
by (*auto simp*: *set-integrable-def integrable-completion*)
have *LBINT x*=*ereal a*..∞. (*b* + *x*) *powr c* = *0* − *?F a* **using** *ab c limits*
by (*intro interval-integral-FTC-nonneg*) (*auto intro*!: *derivative-eq-intros continuous-intros*)
thus *3*: (∫ *x*∈{*a*<..}. ((*b* + *x*) *powr c*) *∂lborel*) = −((*b* + *a*) *powr* (*c* + *1*) / (*c* + *1*))
by (*simp add*: *interval-integral-to-infinity-eq*)
show (*?f has-integral* −((*b* + *a*) *powr* (*c* + *1*) / (*c* + *1*))) {*a*<..}
using *set-borel-integral-eq-integral*[*OF 1*] *3* **by** (*simp add*: *has-integral-iff*)
qed

lemma *fds-converges-altdef2*:
fds-converges f s ⟷ *convergent* (λ*N*. *eval-fds* (*fds-truncate N f*) *s*)
unfolding *fds-converges-def summable-iff-convergent' eval-fds-truncate*
by (*auto simp*: *not-le intro*!: *convergent-cong always-eventually sum.mono-neutral-right*)

lemma *tendsto-eval-fds-truncate*:
assumes *fds-converges f s*
shows (λ*N*. *eval-fds* (*fds-truncate N f*) *s*) ⟶ *eval-fds f s*
proof −
have (λ*N*. *eval-fds* (*fds-truncate N f*) *s*) ⟶ *eval-fds f s* ⟷
(λ*N*. ∑ *i*≤*N*. *fds-nth f i* / *nat-power i s*) ⟶ *eval-fds f s*
unfolding *eval-fds-truncate*
by (*intro filterlim-cong always-eventually allI sum.mono-neutral-left*) (*auto simp*: *not-le*)
also have ... **using** *assms*
by (*simp add*: *fds-converges-iff sums-def' atLeast0AtMost*)
finally show *?thesis* **.**
qed

lemma *linepath-translate-left*: *linepath* (*c* + *a*) (*c* + *a*) = (λ*x*. *c* + *a*) ∘ *linepath a b*
by *auto*

lemma *linepath-translate-right*: *linepath* (*a* + *c*) (*b* + *c*) = (λ*x*. *x* + *c*) ∘ *linepath a b*
by (*auto simp*: *fun-eq-iff linepath-def algebra-simps*)

lemma *has-contour-integral-linepath-same-Im-iff*:
fixes *a b* :: *complex* **and** *f* :: *complex* ⇒ *complex*

assumes *Im a = Im b Re a < Re b*
shows *(f has-contour-integral I) (linepath a b) ⟷*
((λx. f (of-real x + Im a ∗ i)) has-integral I) {Re a..Re b}
proof −
have *deriv*: *vector-derivative ((λx. x − Im a ∗ i) ∘ linepath a b) (at y) = b − a*
for *y*
using *linepath-translate-right*[*of a −Im a ∗ i b*, *symmetric*] **by** *simp*
have *(f has-contour-integral I) (linepath a b) ⟷*
((λx. f (x + Im a ∗ i)) has-contour-integral I) (linepath (a − Im a ∗ i) (b
− Im a ∗ i))
using *linepath-translate-right*[*of a −Im a ∗ i b*] *deriv* **by** (*simp add*: *has-contour-integral*)
also have *. . . ⟷ ((λx. f (x + Im a ∗ i)) has-integral I) {Re a..Re b}* **using**
assms
by (*subst has-contour-integral-linepath-Reals-iff*) (*auto simp*: *complex-is-Real-iff*)
finally show *?thesis* **.**
qed

lemma *contour-integrable-linepath-same-Im-iff*:
fixes *a b* :: *complex* **and** *f* :: *complex ⇒ complex*
assumes *Im a = Im b Re a < Re b*
shows *(f contour-integrable-on linepath a b) ⟷*
(λx. f (of-real x + Im a ∗ i)) integrable-on {Re a..Re b}
using *contour-integrable-on-def has-contour-integral-linepath-same-Im-iff*[*OF assms*]
by *blast*

lemma *contour-integral-linepath-same-Im*:
fixes *a b* :: *complex* **and** *f* :: *complex ⇒ complex*
assumes *Im a = Im b Re a < Re b*
shows *contour-integral (linepath a b) f = integral {Re a..Re b} (λx. f (x + Im*
a ∗ i))
proof (*cases f contour-integrable-on linepath a b*)
case *True*
thus *?thesis* **using** *has-contour-integral-linepath-same-Im-iff*[*OF assms*, *of f*]
using *has-contour-integral-integral has-contour-integral-unique* **by** *blast*
next
case *False*
thus *?thesis* **using** *contour-integrable-linepath-same-Im-iff*[*OF assms*, *of f*]
by (*simp add*: *not-integrable-contour-integral not-integrable-integral*)
qed

lemmas [*simp del*] = *div-mult-self3 div-mult-self4 div-mult-self2 div-mult-self1*

interpretation *cis*: *periodic-fun-simple cis 2 ∗ pi*
by *standard* (*simp-all add*: *complex-eq-iff*)

lemma *analytic-onE-box*:
assumes *f analytic-on A s ∈ A*
obtains *a b* **where** *Re a < Re b Im a < Im b s ∈ box a b f analytic-on box a b*

proof −
from *assms* **obtain** *r* **where** *r*: *r* > *0 f holomorphic-on ball s r*
by (*auto simp*: *analytic-on-def*)
with *open-contains-box*[*of ball s r s*] **obtain** *a b*
where *box a b* ⊆ *ball s r s* ∈ *box a b* ∀ *i*∈*Basis*. *a* · *i* < *b* · *i* **by** *auto*
moreover from *r* **have** *f analytic-on ball s r* **by** (*simp add*: *analytic-on-open*)
ultimately show *?thesis* **using** *that*[*of a b*] *analytic-on-subset*[*of - ball s r box a b*]
by (*auto simp*: *Basis-complex-def*)
qed

lemma *Re-image-box*:
assumes *Re a* < *Re b Im a* < *Im b*
shows *Re* ‘ *box a b* = {*Re a*<..<*Re b*}
using *inner-image-box*[*of 1*::*complex a b*] *assms* **by** (*auto simp*: *Basis-complex-def*)

lemma *Im-image-box*:
assumes *Re a* < *Re b Im a* < *Im b*
shows *Im* ‘ *box a b* = {*Im a*<..<*Im b*}
using *inner-image-box*[*of* i::*complex a b*] *assms* **by** (*auto simp*: *Basis-complex-def*)

lemma *Re-image-cbox*:
assumes *Re a* ≤ *Re b Im a* ≤ *Im b*
shows *Re* ‘ *cbox a b* = {*Re a*..*Re b*}
using *inner-image-cbox*[*of 1*::*complex a b*] *assms* **by** (*auto simp*: *Basis-complex-def*)

lemma *Im-image-cbox*:
assumes *Re a* ≤ *Re b Im a* ≤ *Im b*
shows *Im* ‘ *cbox a b* = {*Im a*..*Im b*}
using *inner-image-cbox*[*of* i::*complex a b*] *assms* **by** (*auto simp*: *Basis-complex-def*)

lemma *analytic-onE-cball*:
assumes *f analytic-on A s* ∈ *A ub* > (*0*::*real*)
obtains *R* **where** *R* > *0 R* < *ub f analytic-on cball s R*
proof −
from *assms* **obtain** *r* **where** *r* > *0 f holomorphic-on ball s r*
by (*auto simp*: *analytic-on-def*)
hence *f analytic-on ball s r* **by** (*simp add*: *analytic-on-open*)
hence *f analytic-on cball s* (*min* (*ub* / *2*) (*r* / *2*))
by (*rule analytic-on-subset*, *subst cball-subset-ball-iff*) (*use* ‹*r* > *0*› **in** *auto*)
moreover have *min* (*ub* / *2*) (*r* / *2*) > *0* **and** *min* (*ub* / *2*) (*r* / *2*) < *ub*
using ‹*r* > *0*› **and** ‹*ub* > *0*› **by** (*auto simp*: *min-def*)
ultimately show *?thesis* **using** *that*[*of min* (*ub* / *2*) (*r* / *2*)]
by *blast*
qed

corollary *analytic-pre-zeta′* [*analytic-intros*]:
assumes *f analytic-on A a* > *0*

shows $(\lambda x.\ \textit{pre-zeta}\ a\ (f\ x))\ \textit{analytic-on}\ A$
using *analytic-on-compose-gen*[*OF assms*(*1*) *analytic-pre-zeta*[*of a UNIV*]] *assms*(*2*)
by (*auto simp*: *o-def*)

corollary *analytic-hurwitz-zeta'* [*analytic-intros*]:
assumes $f\ \textit{analytic-on}\ A\ (\bigwedge x.\ x \in A \Longrightarrow f\ x \neq 1)\ a > 0$
shows $(\lambda x.\ \textit{hurwitz-zeta}\ a\ (f\ x))\ \textit{analytic-on}\ A$
using *analytic-on-compose-gen*[*OF assms*(*1*) *analytic-hurwitz-zeta*[*of a* $-\{1\}$]]
assms(*2*,*3*)
by (*auto simp*: *o-def*)

corollary *analytic-zeta'* [*analytic-intros*]:
assumes $f\ \textit{analytic-on}\ A\ (\bigwedge x.\ x \in A \Longrightarrow f\ x \neq 1)$
shows $(\lambda x.\ \textit{zeta}\ (f\ x))\ \textit{analytic-on}\ A$
using *analytic-on-compose-gen*[*OF assms*(*1*) *analytic-zeta*[*of* $-\{1\}$]] *assms*(*2*)
by (*auto simp*: *o-def*)

lemma *logderiv-zeta-analytic*: $(\lambda s.\ \textit{deriv zeta}\ s\ /\ \textit{zeta}\ s)\ \textit{analytic-on}\ \{s.\ \textit{Re}\ s \geq 1\} - \{1\}$
using *zeta-Re-ge-1-nonzero* **by** (*auto intro*!: *analytic-intros*)

lemma *mult-real-sqrt*: $x \geq 0 \Longrightarrow x * \textit{sqrt}\ y = \textit{sqrt}\ (x\ \hat{}\ 2 * y)$
by (*simp add*: *real-sqrt-mult*)

lemma *arcsin-pos*: $x \in \{0<..1\} \Longrightarrow \textit{arcsin}\ x > 0$
using *arcsin-less-arcsin*[*of 0 x*] **by** *simp*

lemmas *analytic-imp-holomorphic'* = *holomorphic-on-subset*[*OF analytic-imp-holomorphic*]

lemma *residue-simple'*:
assumes *open s* $0 \in s$ *f holomorphic-on s*
shows *residue* $(\lambda w.\ f\ w\ /\ w)\ 0 = f\ 0$
using *residue-simple*[*of s 0 f*] *assms* **by** *simp*

lemma *fds-converges-cong*:
assumes *eventually* $(\lambda n.\ \textit{fds-nth}\ f\ n = \textit{fds-nth}\ g\ n)\ \textit{at-top}\ s = s'$
shows *fds-converges f s* $\longleftrightarrow$ *fds-converges g s'*
unfolding *fds-converges-def*
by (*intro summable-cong eventually-mono*[*OF assms*(*1*)]) (*simp-all add*: *assms*)

lemma *fds-abs-converges-cong*:
assumes *eventually* $(\lambda n.\ \textit{fds-nth}\ f\ n = \textit{fds-nth}\ g\ n)\ \textit{at-top}\ s = s'$
shows *fds-abs-converges f s* $\longleftrightarrow$ *fds-abs-converges g s'*
unfolding *fds-abs-converges-def*
by (*intro summable-cong eventually-mono*[*OF assms*(*1*)]) (*simp-all add*: *assms*)

lemma *conv-abscissa-cong*:

assumes *eventually* (λn. *fds-nth f n* = *fds-nth g n*) *at-top*
shows *conv-abscissa f* = *conv-abscissa g*
proof −
have *fds-converges f* = *fds-converges g*
by (*intro ext fds-converges-cong assms refl*)
thus *?thesis* **by** (*simp add*: *conv-abscissa-def*)
qed

lemma *abs-conv-abscissa-cong*:
assumes *eventually* (λn. *fds-nth f n* = *fds-nth g n*) *at-top*
shows *abs-conv-abscissa f* = *abs-conv-abscissa g*
proof −
have *fds-abs-converges f* = *fds-abs-converges g*
by (*intro ext fds-abs-converges-cong assms refl*)
thus *?thesis* **by** (*simp add*: *abs-conv-abscissa-def*)
qed

definition *fds-remainder* **where**
fds-remainder m = *fds-subseries* (λn. *n* > *m*)

lemma *fds-nth-remainder*: *fds-nth* (*fds-remainder m f*) = (λn. *if n* > *m then fds-nth f n else 0*)
by (*simp add*: *fds-remainder-def fds-subseries-def fds-nth-fds'*)

lemma *fds-converges-remainder-iff* [*simp*]:
fds-converges (*fds-remainder m f*) *s* ⟷ *fds-converges f s*
by (*intro fds-converges-cong eventually-mono*[*OF eventually-gt-at-top*[*of m*]])
(*auto simp*: *fds-nth-remainder*)

lemma *fds-abs-converges-remainder-iff* [*simp*]:
fds-abs-converges (*fds-remainder m f*) *s* ⟷ *fds-abs-converges f s*
by (*intro fds-abs-converges-cong eventually-mono*[*OF eventually-gt-at-top*[*of m*]])
(*auto simp*: *fds-nth-remainder*)

lemma *fds-converges-remainder* [*intro*]:
fds-converges f s ⟹ *fds-converges* (*fds-remainder m f*) *s*
and *fds-abs-converges-remainder* [*intro*]:
fds-abs-converges f s ⟹ *fds-abs-converges* (*fds-remainder m f*) *s*
by *simp-all*

lemma *conv-abscissa-remainder* [*simp*]:
conv-abscissa (*fds-remainder m f*) = *conv-abscissa f*
by (*intro conv-abscissa-cong eventually-mono*[*OF eventually-gt-at-top*[*of m*]])
(*auto simp*: *fds-nth-remainder*)

lemma *abs-conv-abscissa-remainder* [*simp*]:
abs-conv-abscissa (*fds-remainder m f*) = *abs-conv-abscissa f*
by (*intro abs-conv-abscissa-cong eventually-mono*[*OF eventually-gt-at-top*[*of m*]])

(*auto simp*: *fds-nth-remainder*)

lemma *eval-fds-remainder*:
eval-fds (*fds-remainder m f*) *s* = (∑ *n*. *fds-nth f* (*n* + *Suc m*) / *nat-power* (*n* + *Suc m*) *s*)
(**is** - = *suminf* (λ*n*. *?f* (*n* + *Suc m*)))
proof (*cases fds-converges f s*)
case *False*
hence ¬*fds-converges* (*fds-remainder m f*) *s* **by** *simp*
hence (λ*x*. (λ*n*. *fds-nth* (*fds-remainder m f*) *n* / *nat-power n s*) *sums x*) = (λ-. *False*)
by (*auto simp*: *fds-converges-def summable-def*)
hence *eval-fds* (*fds-remainder m f*) *s* = (*THE* -. *False*)
by (*simp add*: *eval-fds-def suminf-def*)
moreover from *False* **have** ¬*summable* (λ*n*. *?f* (*n* + *Suc m*)) **unfolding** *fds-converges-def*
by (*subst summable-iff-shift*) *auto*
hence (λ*x*. (λ*n*. *?f* (*n* + *Suc m*)) *sums x*) = (λ-. *False*)
by (*auto simp*: *summable-def*)
hence *suminf* (λ*n*. *?f* (*n* + *Suc m*)) = (*THE* -. *False*)
by (*simp add*: *suminf-def*)
ultimately show *?thesis* **by** *simp*
next
case *True*
hence ∗: *fds-converges* (*fds-remainder m f*) *s* **by** *simp*
have *eval-fds* (*fds-remainder m f*) *s* = (∑ *n*. *fds-nth* (*fds-remainder m f*) *n* / *nat-power n s*)
unfolding *eval-fds-def* **..**
also have ... = (∑ *n*. *fds-nth* (*fds-remainder m f*) (*n* + *Suc m*) / *nat-power* (*n* + *Suc m*) *s*)
using ∗ **unfolding** *fds-converges-def*
by (*subst suminf-minus-initial-segment*) (*auto simp*: *fds-nth-remainder*)
also have (λ*n*. *fds-nth* (*fds-remainder m f*) (*n* + *Suc m*)) = (λ*n*. *fds-nth f* (*n* + *Suc m*))
by (*intro ext*) (*auto simp*: *fds-nth-remainder*)
finally show *?thesis* **.**
qed

lemma *fds-truncate-plus-remainder*: *fds-truncate m f* + *fds-remainder m f* = *f*
by (*intro fds-eqI*) (*auto simp*: *fds-truncate-def fds-remainder-def fds-subseries-def*)

lemma *holomorphic-fds-eval′* [*holomorphic-intros*]:
assumes *g holomorphic-on A* ⋀*x*. *x* ∈ *A* ⟹ *Re* (*g x*) > *conv-abscissa f*
shows (λ*x*. *eval-fds f* (*g x*)) *holomorphic-on A*
using *holomorphic-on-compose-gen*[*OF assms*(*1*) *holomorphic-fds-eval*[*OF order.refl*, *of f*]] *assms*(*2*)
by (*auto simp*: *o-def*)

lemma *analytic-fds-eval′* [*analytic-intros*]:
assumes *g analytic-on A* $\bigwedge x.\ x \in A \Longrightarrow Re\ (g\ x) >$ *conv-abscissa f*
shows $(\lambda x.$ *eval-fds f* $(g\ x))$ *analytic-on A*
using *analytic-on-compose-gen*[*OF assms*(*1*) *analytic-fds-eval*[*OF order.refl*, *of f*]] *assms*(*2*)
by (*auto simp*: *o-def*)

lemma *continuous-on-linepath* [*continuous-intros*]:
assumes *continuous-on A a continuous-on A b continuous-on A f*
shows *continuous-on A* $(\lambda x.$ *linepath* $(a\ x)\ (b\ x)\ (f\ x))$
using *assms* **by** (*auto simp*: *linepath-def intro*!: *continuous-intros assms*)

lemma *continuous-on-part-circlepath* [*continuous-intros*]:
assumes *continuous-on A c continuous-on A r continuous-on A a continuous-on A b*
continuous-on A f
shows *continuous-on A* $(\lambda x.$ *part-circlepath* $(c\ x)\ (r\ x)\ (a\ x)\ (b\ x)\ (f\ x))$
using *assms* **by** (*auto simp*: *part-circlepath-def intro*!: *continuous-intros assms*)

lemma *homotopic-loops-part-circlepath*:
assumes *sphere c r* $\subseteq$ *A* **and** $r \geq 0$ **and**
b1 $=$ *a1* $+$ *2* $*$ *of-int k* $*$ *pi* **and** *b2* $=$ *a2* $+$ *2* $*$ *of-int k* $*$ *pi*
shows *homotopic-loops A* (*part-circlepath c r a1 b1*) (*part-circlepath c r a2 b2*)
proof −
define *h* **where** $h = (\lambda(x,y).$ *part-circlepath c r* (*linepath a1 a2 x*) (*linepath b1 b2 x*) *y*)
show *?thesis*
proof (*rule homotopic-loopsI*)
show *continuous-on* $(\{0..1\} \times \{0..1\})$ *h*
by (*auto simp*: *h-def case-prod-unfold intro*!: *continuous-intros*)
next
from *assms* **have** *h* ‘ $(\{0..1\} \times \{0..1\}) \subseteq$ *sphere c r*
by (*auto simp*: *h-def part-circlepath-def dist-norm norm-mult*)
also have $\ldots \subseteq$ *A* **by** *fact*
finally show *h* ‘ $(\{0..1\} \times \{0..1\}) \subseteq$ *A* **.**
next
fix *x* :: *real* **assume** *x*: $x \in \{0..1\}$
show *h* $(0,\ x) =$ *part-circlepath c r a1 b1 x* **and** *h* $(1,\ x) =$ *part-circlepath c r a2 b2 x*
by (*simp-all add*: *h-def linepath-def*)
have *cis* $(pi * (real\text{-}of\text{-}int\ k * 2)) = 1$
using *cis.plus-of-int*[*of 0 k*] **by** (*simp add*: *algebra-simps*)
thus *pathfinish* $(h \circ Pair\ x) =$ *pathstart* $(h \circ Pair\ x)$
by (*simp add*: *h-def o-def exp-eq-polar linepath-def algebra-simps*
cis-mult [*symmetric*] *cis-divide* [*symmetric*] *assms*)
qed
qed

lemma *part-circlepath-conv-subpath*:

part-circlepath c r a b = subpath (a / (2∗pi)) (b / (2∗pi)) (circlepath c r)
by (*simp add*: *part-circlepath-def circlepath-def subpath-def linepath-def algebra-simps exp-eq-polar*)

lemma *homotopic-paths-part-circlepath*:
assumes $a \leq b$ $b \leq c$
assumes *path-image* (*part-circlepath C r a c*) $\subseteq A$ $r \geq 0$
shows *homotopic-paths A* (*part-circlepath C r a c*)
(*part-circlepath C r a b +++ part-circlepath C r b c*)
(**is** *homotopic-paths - ?g* (*?h1 +++ ?h2*))
proof (*cases a = c*)
case *False*
with *assms* **have** $a < c$ **by** *simp*
define *slope* **where** *slope* = (*b − a*) / (*c − a*)
from *assms* **and** ‹*a < c*› **have** *slope*: *slope* ∈ {*0..1*}
by (*auto simp*: *field-simps slope-def*)
define *f* :: *real* ⇒ *real* **where**
f = linepath 0 slope +++ linepath slope 1

show *?thesis*
proof (*rule homotopic-paths-reparametrize*)
fix *t* :: *real* **assume** *t*: *t* ∈ {*0..1*}
show (*?h1 +++ ?h2*) *t = ?g* (*f t*)
proof (*cases t ≤ 1 / 2*)
case *True*
hence *?g* (*f t*) = *C + r ∗ cis* ((*1 − f t*) ∗ *a + f t ∗ c*)
by (*simp add*: *joinpaths-def part-circlepath-def exp-eq-polar linepath-def*)
also from *True* ‹*a < c*› **have** (*1 − f t*) ∗ *a + f t ∗ c = (1 − 2 ∗ t*) ∗ *a + 2 ∗ t ∗ b*
unfolding *f-def slope-def linepath-def joinpaths-def*
by (*simp add*: *divide-simps del*: *div-mult-self3 div-mult-self4 div-mult-self2 div-mult-self1*)
(*simp add*: *algebra-simps*)*?*
also from *True* **have** *C + r ∗ cis ... = (?h1 +++ ?h2) t*
by (*simp add*: *joinpaths-def part-circlepath-def exp-eq-polar linepath-def*)
finally show *?thesis* **..**
next
case *False*
hence *?g* (*f t*) = *C + r ∗ cis* ((*1 − f t*) ∗ *a + f t ∗ c*)
by (*simp add*: *joinpaths-def part-circlepath-def exp-eq-polar linepath-def*)
also from *False* ‹*a < c*› **have** (*1 − f t*) ∗ *a + f t ∗ c = (2 − 2 ∗ t*) ∗ *b +* (*2 ∗ t − 1*) ∗ *c*
unfolding *f-def slope-def linepath-def joinpaths-def*
by (*simp add*: *divide-simps del*: *div-mult-self3 div-mult-self4 div-mult-self2 div-mult-self1*)
(*simp add*: *algebra-simps*)*?*
also from *False* **have** *C + r ∗ cis ... = (?h1 +++ ?h2) t*
by (*simp add*: *joinpaths-def part-circlepath-def exp-eq-polar linepath-def*)
finally show *?thesis* **..**

```
    qed
  next
    from slope have path-image f ⊆ {0..1}
      by (auto simp: f-def path-image-join closed-segment-eq-real-ivl)
    thus f ∈ {0..1} → {0..1} by (force simp add: path-image-def)
  next
    have path f unfolding f-def by auto
    thus continuous-on {0..1} f by (simp add: path-def)
  qed (insert assms, auto simp: f-def joinpaths-def linepath-def)
next
  case [simp]: True
  with assms have [simp]: b = c by auto
  have part-circlepath C r c c +++ part-circlepath C r c c = part-circlepath C r c
c
    by (simp add: fun-eq-iff joinpaths-def part-circlepath-def)
  thus ?thesis using assms by simp
qed

lemma path-image-part-circlepath-subset:
  assumes a ≤ a' a' ≤ b' b' ≤ b
  shows   path-image (part-circlepath c r a' b') ⊆ path-image (part-circlepath c r
a b)
  using assms by (subst (1 2) path-image-part-circlepath) auto

lemma part-circlepath-mirror:
  assumes a' = a + pi + 2 * pi * of-int k b' = b + pi + 2 * pi * of-int k c' =
−c
  shows   −part-circlepath c r a b = part-circlepath c' r a' b'
proof
  fix x :: real
  have part-circlepath c' r a' b' x = c' + r * cis (linepath a b x + pi + k * (2 *
pi))
     by (simp add: part-circlepath-def exp-eq-polar assms linepath-translate-right
mult-ac)
  also have cis (linepath a b x + pi + k * (2 * pi)) = cis (linepath a b x + pi)
    by (rule cis.plus-of-int)
  also have ... = −cis (linepath a b x)
    by (simp add: minus-cis)
  also have c' + r * ... = −part-circlepath c r a b x
    by (simp add: part-circlepath-def assms exp-eq-polar)
  finally show (− part-circlepath c r a b) x = part-circlepath c' r a' b' x
    by simp
qed

lemma path-mirror [intro]: path (g :: - ⇒ 'b::topological-group-add) ⟹ path (−g)
  by (auto simp: path-def intro!: continuous-intros)

lemma path-mirror-iff [simp]: path (−g :: - ⇒ 'b::topological-group-add) ⟷ path
g
```

using *path-mirror*[*of g*] *path-mirror*[*of* $-g$] **by** (*auto simp*: *fun-Compl-def*)

lemma *valid-path-mirror* [*intro*]: *valid-path g* $\Longrightarrow$ *valid-path* $(-g)$
by (*auto simp*: *valid-path-def fun-Compl-def piecewise-C1-differentiable-neg*)

lemma *valid-path-mirror-iff* [*simp*]: *valid-path* $(-g) \longleftrightarrow$ *valid-path g*
using *valid-path-mirror*[*of g*] *valid-path-mirror*[*of* $-g$] **by** (*auto simp*: *fun-Compl-def*)

lemma *pathstart-mirror* [*simp*]: *pathstart* $(-g) = -$*pathstart g*
and *pathfinish-mirror* [*simp*]: *pathfinish* $(-g) = -$*pathfinish g*
by (*simp-all add*: *pathstart-def pathfinish-def*)

lemma *path-image-mirror*: *path-image* $(-g) =$ *uminus* ‘ *path-image g*
by (*auto simp*: *path-image-def*)

lemma *cos-le-zero*:
assumes $x \in \{pi/2..3*pi/2\}$
shows *cos x* ≤ 0
proof −
have *cos x* $= -$*cos* $(x - pi)$ **by** (*simp add*: *cos-diff*)
moreover from *assms* **have** *cos* $(x - pi) \geq 0$
by (*intro cos-ge-zero*) *auto*
ultimately show *?thesis* **by** *simp*
qed

lemma *cos-le-zero′*: $x \in \{-3*pi/2..-pi/2\} \Longrightarrow$ *cos x* ≤ 0
using *cos-le-zero*[*of* $-x$] **by** *simp*

lemma *winding-number-join-pos-combined′*:
⟦*valid-path* $\gamma 1 \wedge z \notin$ *path-image* $\gamma 1 \wedge 0 <$ *Re* (*winding-number* $\gamma 1$ *z*);
valid-path $\gamma 2 \wedge z \notin$ *path-image* $\gamma 2 \wedge 0 <$ *Re* (*winding-number* $\gamma 2$ *z*);
pathfinish $\gamma 1 =$ *pathstart* $\gamma 2$⟧
$\Longrightarrow$ *valid-path*($\gamma 1$ +++ $\gamma 2$) $\wedge$ *z* $\notin$ *path-image*($\gamma 1$ +++ $\gamma 2$) $\wedge$ 0 $<$ *Re*(*winding-number*($\gamma 1$ +++ $\gamma 2$) *z*)
by (*simp add*: *valid-path-join path-image-join winding-number-join valid-path-imp-path*)

lemma *Union-atLeastAtMost-real-of-nat*:
assumes $a < b$
shows $(\bigcup n{\in}\{a..<b\}.\ \{real\ n..real\ (n + 1)\}) = \{real\ a..real\ b\}$
proof (*intro equalityI subsetI*)
fix *x* **assume** *x*: $x \in \{real\ a..real\ b\}$
thus $x \in (\bigcup n{\in}\{a..<b\}.\ \{real\ n..real\ (n + 1)\})$
proof (*cases x* = *real b*)
case *True*
with *assms* **show** *?thesis* **by** (*auto intro*!: *bexI*[*of* - *b* − *1*])
next
case *False*
with *x* **have** *x*: $x \geq real\ a\ x < real\ b$ **by** *simp-all*
hence $x \geq real\ (nat\ \lfloor x \rfloor)\ x \leq real\ (Suc\ (nat\ \lfloor x \rfloor))$ **by** *linarith*+

```
    moreover from x have nat ⌊x⌋ ≥ a nat ⌊x⌋ < b by linarith+
    ultimately show ?thesis by force
  qed
qed auto

lemma nat-sum-has-integral-floor:
  fixes f :: nat ⇒ 'a :: banach
  assumes mn: m < n
  shows ((λx. f (nat ⌊x⌋)) has-integral sum f {m..<n}) {real m..real n}
proof −
  define D where D = (λi. {real i..real (Suc i)}) ` {m..<n}
  have D: D division-of {m..n}
    using Union-atLeastAtMost-real-of-nat[OF mn] by (simp add: division-of-def
D-def)
  have ((λx. f (nat ⌊x⌋)) has-integral (∑ X∈D. f (nat ⌊Inf X⌋))) {real m..real n}
  proof (rule has-integral-combine-division)
    fix X assume X: X ∈ D
    have nat ⌊x⌋ = nat ⌊Inf X⌋ if x ∈ X − {Sup X} for x
      using that X by (auto simp: D-def nat-eq-iff floor-eq-iff)
    hence ((λx. f (nat ⌊x⌋)) has-integral f (nat ⌊Inf X⌋)) X ⟷
          ((λx. f (nat ⌊Inf X⌋)) has-integral f (nat ⌊Inf X⌋)) X using X
      by (intro has-integral-spike-eq[of {Sup X}]) auto
    also from X have ... using has-integral-const-real[of f (nat ⌊Inf X⌋) Inf X
Sup X]
      by (auto simp: D-def)
    finally show ((λx. f (nat ⌊x⌋)) has-integral f (nat ⌊Inf X⌋)) X .
  qed fact+
  also have (∑ X∈D. f (nat ⌊Inf X⌋)) = (∑ k∈{m..<n}. f k)
    unfolding D-def by (subst sum.reindex) (auto simp: inj-on-def nat-add-distrib)
  finally show ?thesis .
qed

lemma nat-sum-has-integral-ceiling:
  fixes f :: nat ⇒ 'a :: banach
  assumes mn: m < n
  shows ((λx. f (nat ⌈x⌉)) has-integral sum f {m<..n}) {real m..real n}
proof −
  define D where D = (λi. {real i..real (Suc i)}) ` {m..<n}
  have D: D division-of {m..n}
    using Union-atLeastAtMost-real-of-nat[OF mn] by (simp add: division-of-def
D-def)
  have ((λx. f (nat ⌈x⌉)) has-integral (∑ X∈D. f (nat ⌊Sup X⌋))) {real m..real n}
  proof (rule has-integral-combine-division)
    fix X assume X: X ∈ D
    have nat ⌈x⌉ = nat ⌊Sup X⌋ if x ∈ X − {Inf X} for x
      using that X by (auto simp: D-def nat-eq-iff ceiling-eq-iff)
    hence ((λx. f (nat ⌈x⌉)) has-integral f (nat ⌊Sup X⌋)) X ⟷
          ((λx. f (nat ⌊Sup X⌋)) has-integral f (nat ⌊Sup X⌋)) X using X
      by (intro has-integral-spike-eq[of {Inf X}]) auto
```

```
    also from X have ... using has-integral-const-real[of f (nat ⌊Sup X⌋) Inf X
Sup X]
      by (auto simp: D-def)
    finally show ((λx. f (nat ⌈x⌉)) has-integral f (nat ⌊Sup X⌋)) X .
  qed fact+
  also have (∑ X∈D. f (nat ⌊Sup X⌋)) = (∑ k∈{m..<n}. f (Suc k))
    unfolding D-def by (subst sum.reindex) (auto simp: inj-on-def nat-add-distrib)
  also have ... = (∑ k∈{m<..n}. f k)
    by (intro sum.reindex-bij-witness[of - λx. x − 1 Suc]) auto
  finally show ?thesis .
qed

lemma zeta-partial-sum-le:
  fixes x :: real and m :: nat
  assumes x: x ∈ {0<..1}
  shows (∑ k=1..m. real k powr (x − 1)) ≤ real m powr x / x
proof −
  consider m = 0 | m = 1 | m > 1 by force
  thus ?thesis
  proof cases
    assume m: m > 1
    hence {1..m} = insert 1 {1<..m} by auto
    also have (∑ k∈.... real k powr (x − 1)) = 1 + (∑ k∈{1<..m}. real k powr
(x − 1))
      by simp
    also have (∑ k∈{1<..m}. real k powr (x − 1)) ≤ real m powr x / x − 1 / x
    proof (rule has-integral-le)
      show ((λt. (nat ⌈t⌉) powr (x − 1)) has-integral (∑ n∈{1<..m}. n powr (x
− 1))) {real 1..m}
        using m by (intro nat-sum-has-integral-ceiling) auto
    next
      have ((λt. t powr (x − 1)) has-integral (real m powr x / x − real 1 powr x /
x))
             {real 1..real m}
        by (intro fundamental-theorem-of-calculus)
           (insert x m, auto simp flip: has-real-derivative-iff-has-vector-derivative
                        intro!: derivative-eq-intros)
      thus ((λt. t powr (x − 1)) has-integral (real m powr x / x − 1 / x)) {real
1..real m}
        by simp
    qed (insert x, auto intro!: powr-mono2')
    also have 1 + (real m powr x / x − 1 / x) ≤ real m powr x / x
      using x by (simp add: field-simps)
    finally show ?thesis by simp
  qed (use assms in auto)
qed

lemma zeta-partial-sum-le':
  fixes x :: real and m :: nat
```

```
  assumes x: x > 0 and m: m > 0
  shows   (∑ n=1..m. real n powr (x − 1)) ≤ m powr x * (1 / x + 1 / m)
proof (cases x > 1)
  case False
  with assms have (∑ n=1..m. real n powr (x − 1)) ≤ m powr x / x
    by (intro zeta-partial-sum-le) auto
  also have ... ≤ m powr x * (1 / x + 1 / m)
    using assms by (simp add: field-simps)
  finally show ?thesis .
next
  case True
  have (∑ n∈{1..m}. n powr (x − 1)) = (∑ n∈insert m {0..<m}. n powr (x −
1))
    by (intro sum.mono-neutral-left) auto
  also have ... = m powr (x − 1) + (∑ n∈{0..<m}. n powr (x − 1)) by simp
  also have (∑ n∈{0..<m}. n powr (x − 1)) ≤ real m powr x / x
  proof (rule has-integral-le)
    show ((λt. (nat ⌊t⌋) powr (x − 1)) has-integral (∑ n∈{0..<m}. n powr (x −
1))) {real 0..m}
      using m by (intro nat-sum-has-integral-floor) auto
  next
    show ((λt. t powr (x − 1)) has-integral (real m powr x / x)) {real 0..real m}
      using has-integral-powr-from-0[of x − 1] x by auto
  next
    fix t assume t ∈ {real 0..real m}
    with ‹x > 1› show real (nat ⌊t⌋) powr (x − 1) ≤ t powr (x − 1)
      by (cases t = 0) (auto intro: powr-mono2)
  qed
  also have m powr (x − 1) + m powr x / x = m powr x * (1 / x + 1 / m)
    using m x by (simp add: powr-diff field-simps)
  finally show ?thesis by simp
qed

lemma natfun-bigo-1E:
  assumes (f :: nat ⇒ -) ∈ O(λ-. 1)
  obtains C where C ≥ lb ⋀n. norm (f n) ≤ C
proof −
  from assms obtain C N where ∀ n≥N. norm (f n) ≤ C
    by (auto elim!: landau-o.bigE simp: eventually-at-top-linorder)
  hence *: norm (f n) ≤ Max ({C, lb} ∪ (norm ‘ f ‘ {..<N})) for n
    by (cases n ≥ N) (subst Max-ge-iff; force simp: image-iff)+
  moreover have Max ({C, lb} ∪ (norm ‘ f ‘ {..<N})) ≥ lb
    by (intro Max.coboundedI) auto
  ultimately show ?thesis using that by blast
qed

lemma natfun-bigo-iff-Bseq: f ∈ O(λ-. 1) ⟷ Bseq f
proof
  assume Bseq f
```

then obtain C **where** $C > 0$ $\bigwedge n.$ *norm* $(f\ n) \leq C$ **by** (*auto simp*: *Bseq-def*)
thus $f \in O(\lambda\text{-}.\ 1)$ **by** (*intro bigoI*[*of* - C]) *auto*
next
assume $f \in O(\lambda\text{-}.\ 1)$
from *natfun-bigo-1E*[*OF this*, **where** $lb = 1$] **obtain** C **where** $C \geq 1$ $\bigwedge n.$ *norm* $(f\ n) \leq C$
by *auto*
thus *Bseq* f **by** (*auto simp*: *Bseq-def intro*!: *exI*[*of* - C])
qed

lemma *enn-decreasing-sum-le-set-nn-integral*:
fixes f :: *real* $\Rightarrow$ *ennreal*
assumes *decreasing*: $\bigwedge x\ y.\ 0 \leq x \Longrightarrow x \leq y \Longrightarrow f\ y \leq f\ x$
shows $(\sum n.\ f\ (real\ (Suc\ n))) \leq$ *set-nn-integral lborel* $\{0..\}$ f
proof −
have $(\sum n.\ (f\ (Suc\ n))) =$
$(\sum n.\ \int^+ x{\in}\{real\ n{<}..real\ (Suc\ n)\}.\ (f\ (Suc\ n))\ \partial lborel)$
by (*subst nn-integral-cmult-indicator*) *auto*
also have *nat* $\lceil x \rceil$ = *Suc* n **if** $x \in \{real\ n{<}..real\ (Suc\ n)\}$ **for** $x\ n$
using *that* **by** (*auto simp*: *nat-eq-iff ceiling-eq-iff*)
hence $(\sum n.\ \int^+ x{\in}\{real\ n{<}..real\ (Suc\ n)\}.\ (f\ (Suc\ n))\ \partial lborel) =$
$(\sum n.\ \int^+ x{\in}\{real\ n{<}..real\ (Suc\ n)\}.\ (f\ (real\ (nat\ \lceil x \rceil)))\ \partial lborel)$
by (*intro suminf-cong nn-integral-cong*) (*auto simp*: *indicator-def*)
also have ... $= (\int^+ x{\in}(\bigcup i.\ \{real\ i{<}..real\ (Suc\ i)\}).\ (f\ (nat\ \lceil x{::}real \rceil))\ \partial lborel)$
by (*subst nn-integral-disjoint-family*)
(*auto simp*: *disjoint-family-on-def*)
also have ... $\leq (\int^+ x{\in}\{0..\}.\ (f\ x)\ \partial lborel)$
by (*intro nn-integral-mono*) (*auto simp*: *indicator-def intro*!: *decreasing*)
finally show *?thesis* **.**
qed

lemma *abs-summable-on-uminus-iff*:
$(\lambda x.\ -f\ x)$ *abs-summable-on* $A \longleftrightarrow f$ *abs-summable-on* A
by (*simp add*: *abs-summable-on-def*)

lemma *abs-summable-on-cmult-right-iff*:
fixes f :: $'a \Rightarrow 'b$ :: {*banach*, *real-normed-field*, *second-countable-topology*}
assumes $c \neq 0$
shows $(\lambda x.\ c * f\ x)$ *abs-summable-on* $A \longleftrightarrow f$ *abs-summable-on* A
by (*simp add*: *abs-summable-on-altdef assms*)

lemma *abs-summable-on-cmult-left-iff*:
fixes f :: $'a \Rightarrow 'b$ :: {*banach*, *real-normed-field*, *second-countable-topology*}
assumes $c \neq 0$
shows $(\lambda x.\ f\ x * c)$ *abs-summable-on* $A \longleftrightarrow f$ *abs-summable-on* A
by (*simp add*: *abs-summable-on-altdef assms*)

lemma *decreasing-sum-le-integral*:
fixes f :: *real* $\Rightarrow$ *real*

assumes *nonneg*: $\bigwedge x.\ x \geq 0 \Longrightarrow f\ x \geq 0$
assumes *decreasing*: $\bigwedge x\ y.\ 0 \leq x \Longrightarrow x \leq y \Longrightarrow f\ y \leq f\ x$
assumes *integral*: $(f$ *has-integral* $I)\ \{0..\}$
shows *summable* $(\lambda i.\ f\ (real\ (Suc\ i)))$ **and** *suminf* $(\lambda i.\ f\ (real\ (Suc\ i))) \leq I$
proof −
have [*simp*]: $I \geq 0$
by (*intro has-integral-nonneg*[*OF integral*] *nonneg*) *auto*
have $(\sum n.\ ennreal\ (f\ (Suc\ n))) =$
$(\sum n.\ \int^+ x{\in}\{real\ n{<}..real\ (Suc\ n)\}.\ ennreal\ (f\ (Suc\ n))\ \partial lborel)$
by (*subst nn-integral-cmult-indicator*) *auto*
also have *nat* $\lceil x \rceil = Suc\ n$ **if** $x \in \{real\ n{<}..real\ (Suc\ n)\}$ **for** $x\ n$
using *that* **by** (*auto simp*: *nat-eq-iff ceiling-eq-iff*)
hence $(\sum n.\ \int^+ x{\in}\{real\ n{<}..real\ (Suc\ n)\}.\ ennreal\ (f\ (Suc\ n))\ \partial lborel) =$
$(\sum n.\ \int^+ x{\in}\{real\ n{<}..real\ (Suc\ n)\}.\ ennreal\ (f\ (real\ (nat\ \lceil x \rceil)))\ \partial lborel)$
by (*intro suminf-cong nn-integral-cong*) (*auto simp*: *indicator-def*)
also have $\ldots = (\int^+ x{\in}(\bigcup i.\ \{real\ i{<}..real\ (Suc\ i)\}).\ ennreal\ (f\ (nat\ \lceil x{::}real \rceil))\ \partial lborel)$
by (*subst nn-integral-disjoint-family*)
(*auto simp*: *disjoint-family-on-def intro*!: *measurable-completion*)
also have $\ldots \leq (\int^+ x{\in}\{0..\}.\ ennreal\ (f\ x)\ \partial lborel)$
by (*intro nn-integral-mono*) (*auto simp*: *indicator-def nonneg intro*!: *decreasing*)
also have $\ldots = (\int^+ x.\ ennreal\ (indicat\text{-}real\ \{0..\}\ x * f\ x)\ \partial lborel)$
by (*intro nn-integral-cong*) (*auto simp*: *indicator-def*)
also have $\ldots = ennreal\ I$
using *nn-integral-has-integral-lebesgue*[*OF nonneg integral*] **by** (*auto simp*: *nonneg*)
finally have $*$: $(\sum n.\ ennreal\ (f\ (Suc\ n))) \leq ennreal\ I$ **.**
from $*$ **show** *summable*: *summable* $(\lambda i.\ f\ (real\ (Suc\ i)))$
by (*intro summable-suminf-not-top*) (*auto simp*: *top-unique intro*: *nonneg*)
note $*$
also from *summable* **have** $(\sum n.\ ennreal\ (f\ (Suc\ n))) = ennreal\ (\sum n.\ f\ (Suc\ n))$
by (*subst suminf-ennreal2*) (*auto simp*: *o-def nonneg*)
finally show $(\sum n.\ f\ (real\ (Suc\ n))) \leq I$ **by** (*subst* (*asm*) *ennreal-le-iff*) *auto*
qed

lemma *decreasing-sum-le-integral′*:
fixes $f :: real \Rightarrow real$
assumes $\bigwedge x.\ x \geq 0 \Longrightarrow f\ x \geq 0$
assumes $\bigwedge x\ y.\ 0 \leq x \Longrightarrow x \leq y \Longrightarrow f\ y \leq f\ x$
assumes $(f$ *has-integral* $I)\ \{0..\}$
shows *summable* $(\lambda i.\ f\ (real\ i))$ **and** *suminf* $(\lambda i.\ f\ (real\ i)) \leq f\ 0 + I$
proof −
have *summable* $((\lambda i.\ f\ (real\ (Suc\ i))))$
using *decreasing-sum-le-integral*[*OF assms*] **by** (*simp add*: *o-def*)
thus $*$: *summable* $(\lambda i.\ f\ (real\ i))$ **by** (*subst* (*asm*) *summable-Suc-iff*)
have $(\sum n.\ f\ (real\ (Suc\ n))) \leq I$ **by** (*intro decreasing-sum-le-integral assms*)
thus *suminf* $(\lambda i.\ f\ (real\ i)) \leq f\ 0 + I$
using $*$ **by** (*subst* (*asm*) *suminf-split-head*) *auto*

qed

lemma *of-nat-powr-neq-1-complex* [*simp*]:
assumes $n > 1$ *Re* $s \neq 0$
shows *of-nat n powr s* $\neq$ (*1*::*complex*)
proof −
have *norm* (*of-nat n powr s*) = *real n powr Re s*
by (*simp add*: *norm-powr-real-powr*)
also have ... $\neq$ *1*
using *assms* **by** (*auto simp*: *powr-def*)
finally show *?thesis* **by** *auto*
qed

lemma *fds-logderiv-completely-multiplicative*:
fixes *f* :: *'a* :: {*real-normed-field*} *fds*
assumes *completely-multiplicative-function* (*fds-nth f*) *fds-nth f 1* $\neq$ *0*
shows *fds-deriv f / f* = − *fds* (λ*n*. *fds-nth f n* ∗ *mangoldt n*)
proof −
have *fds-deriv f / f* = − *fds* (λ*n*. *fds-nth f n* ∗ *mangoldt n*) ∗ *f / f*
using *completely-multiplicative-fds-deriv*[*of fds-nth f*] *assms* **by** *simp*
also have ... = − *fds* (λ*n*. *fds-nth f n* ∗ *mangoldt n*)
using *assms* **by** (*simp add*: *divide-fds-def fds-right-inverse*)
finally show *?thesis* **.**
qed

lemma *fds-nth-logderiv-completely-multiplicative*:
fixes *f* :: *'a* :: {*real-normed-field*} *fds*
assumes *completely-multiplicative-function* (*fds-nth f*) *fds-nth f 1* $\neq$ *0*
shows *fds-nth* (*fds-deriv f / f*) *n* = −*fds-nth f n* ∗ *mangoldt n*
using *assms* **by** (*subst fds-logderiv-completely-multiplicative*) (*simp-all add*: *fds-nth-fds'*)

lemma *eval-fds-logderiv-completely-multiplicative*:
fixes *s* :: *'a* :: *dirichlet-series* **and** *l* :: *'a* **and** *f* :: *'a fds*
defines *h* ≡ *fds-deriv f / f*
assumes *completely-multiplicative-function* (*fds-nth f*) **and** [*simp*]: *fds-nth f 1* $\neq$ *0*
assumes *s* · *1* > *abs-conv-abscissa f*
shows (λ*p*. *of-real* (*ln* (*real p*)) ∗ (*1* / (*1* − *fds-nth f p* / *nat-power p s*) − *1*))
abs-summable-on {*p*. *prime p*} (**is** *?th1*)
and *eval-fds h s* = −($\sum_a$*p* | *prime p*. *of-real* (*ln* (*real p*)) ∗
(*1* / (*1* − *fds-nth f p* / *nat-power p s*) − *1*)) (**is** *?th2*)
proof −
let *?P* = {*p*::*nat*. *prime p*}
interpret *f*: *completely-multiplicative-function fds-nth f* **by** *fact*
have *fds-abs-converges h s*
using *abs-conv-abscissa-completely-multiplicative-log-deriv*[*OF assms*(*2*)] *assms*
by (*intro fds-abs-converges*) *auto*
hence ∗: (λ*n*. *fds-nth h n* / *nat-power n s*) *abs-summable-on UNIV*
by (*auto simp*: *h-def fds-abs-converges-altdef'*)

```
  note ∗
  also have (λn. fds-nth h n / nat-power n s) abs-summable-on UNIV ⟷
        (λx. −fds-nth f x ∗ mangoldt x / nat-power x s) abs-summable-on Collect
primepow
    unfolding h-def using fds-nth-logderiv-completely-multiplicative[OF assms(2)]
    by (intro abs-summable-on-cong-neutral) (auto simp: fds-nth-fds mangoldt-def)
  finally have sum1: (λx. −fds-nth f x ∗ mangoldt x / nat-power x s)
                  abs-summable-on Collect primepow
    by (rule abs-summable-on-subset) auto
  also have ?this ⟷ (λ(p,k). −fds-nth f (p ^ Suc k) ∗ mangoldt (p ^ Suc k) /
                          nat-power (p ^ Suc k) s) abs-summable-on (?P × UNIV)
    using bij-betw-primepows unfolding case-prod-unfold
    by (intro abs-summable-on-reindex-bij-betw [symmetric])
  also have ... ⟷ (λ(p,k). −((fds-nth f p / nat-power p s) ^ Suc k ∗ of-real (ln
(real p))))
              abs-summable-on (?P × UNIV)
    unfolding case-prod-unfold
    by (intro abs-summable-on-cong, subst mangoldt-primepow)
     (auto simp: f.mult f.power nat-power-mult-distrib nat-power-power-left power-divide
           dest: prime-gt-1-nat)
  finally have sum2: ... .

  have sum4: summable (λn. (norm (fds-nth f p / nat-power p s)) ^ Suc n) if p:
prime p for p
  proof −
    have summable (λn. |ln (real p)| ∗ (norm (fds-nth f p / nat-power p s)) ^ Suc
n)
      using p abs-summable-on-Sigma-project2[OF sum2, of p] unfolding abs-summable-on-nat-iff'
      by (simp add: norm-power norm-mult norm-divide mult-ac del: power-Suc)
    thus ?thesis by (rule summable-mult-D) (insert p, auto dest: prime-gt-1-nat)
  qed
  have sums: (λn. (fds-nth f p / nat-power p s) ^ Suc n) sums
              (1 / (1 − fds-nth f p / nat-power p s) − 1) if p: prime p for p :: nat
  proof −
    from sum4[OF p] have norm (fds-nth f p / nat-power p s) < 1
      unfolding summable-Suc-iff by (simp add: summable-geometric-iff)
    from geometric-sums[OF this] show ?thesis by (subst sums-Suc-iff) auto
  qed

  have eq: (∑a k. − ((fds-nth f p / nat-power p s) ^ Suc k ∗ of-real (ln (real p))))
=
              −(of-real (ln (real p)) ∗ (1 / (1 − fds-nth f p / nat-power p s) − 1))
    if p: prime p for p
  proof −
    have (∑a k. − ((fds-nth f p / nat-power p s) ^ Suc k ∗ of-real (ln (real p)))) =
          (∑a k. (fds-nth f p / nat-power p s) ^ Suc k) ∗ of-real (−ln (real p))
      using sum4[of p] p
      by (subst infsetsum-cmult-left [symmetric])
```

```
      (auto simp: abs-summable-on-nat-iff' norm-power simp del: power-Suc)
  also have (∑a k. (fds-nth f p / nat-power p s) ^ Suc k) =
              (1 / (1 - fds-nth f p / nat-power p s) - 1) using sum4[OF p]
sums[OF p]
    by (subst infsetsum-nat') 
       (auto simp: sums-iff abs-summable-on-nat-iff' norm-power simp del:
power-Suc)
  finally show ?thesis by (simp add: mult-ac)
 qed

 have sum3: (λx. ∑a y. - ((fds-nth f x / nat-power x s) ^ Suc y * of-real (ln
(real x))))
            abs-summable-on {p. prime p}
  using sum2 by (rule abs-summable-on-Sigma-project1') auto
 also have ?this ⟷ (λp. -(of-real (ln (real p)) *
            (1 / (1 - fds-nth f p / nat-power p s) - 1))) abs-summable-on {p.
prime p}
  by (intro abs-summable-on-cong eq) auto
 also have ... ⟷ ?th1 by (subst abs-summable-on-uminus-iff) auto
 finally show ?th1 .

 have eval-fds h s = (∑a n. fds-nth h n / nat-power n s)
  using * unfolding eval-fds-def by (subst infsetsum-nat') auto
 also have ... = (∑a n ∈ {n. primepow n}. -fds-nth f n * mangoldt n / nat-power
n s)
  unfolding h-def using fds-nth-logderiv-completely-multiplicative[OF assms(2)]
  by (intro infsetsum-cong-neutral) (auto simp: fds-nth-fds mangoldt-def)
 also have ... = (∑a (p,k)∈(?P × UNIV). -fds-nth f (p ^ Suc k) * mangoldt
(p ^ Suc k) /
                                    nat-power (p ^ Suc k) s)
   using bij-betw-primepows unfolding case-prod-unfold
   by (intro infsetsum-reindex-bij-betw [symmetric])
 also have ... = (∑a (p,k)∈(?P × UNIV).
              -((fds-nth f p / nat-power p s) ^ Suc k) * of-real (ln (real p)))
  by (intro infsetsum-cong)
    (auto simp: f.mult f.power mangoldt-def aprimedivisor-prime-power ln-realpow
prime-gt-0-nat
       nat-power-power-left divide-simps simp del: power-Suc)
 also have ... = (∑a p | prime p. ∑a k.
              - ((fds-nth f p / nat-power p s) ^ Suc k) * of-real (ln (real p)))
  using sum2 by (subst infsetsum-Times) (auto simp: case-prod-unfold)
 also have ... = (∑a p | prime p. -(of-real (ln (real p)) *
              (1 / (1 - fds-nth f p / nat-power p s) - 1)))
  using eq by (intro infsetsum-cong) auto
 finally show ?th2 by (subst (asm) infsetsum-uminus)
qed

lemma eval-fds-logderiv-zeta:
 assumes Re s > 1
```

shows $(\lambda p.\ of\text{-}real\ (ln\ (real\ p))\ /\ (p\ powr\ s - 1))$
$abs\text{-}summable\text{-}on\ \{p.\ prime\ p\}$ (**is** *?th1*)
and $deriv\ zeta\ s\ /\ zeta\ s =$
$-(\sum_a p \mid prime\ p.\ of\text{-}real\ (ln\ (real\ p))\ /\ (p\ powr\ s - 1))$ (**is** *?th2*)
proof −
have $*$: *completely-multiplicative-function* (*fds-nth fds-zeta* :: - ⇒ *complex*)
by *standard auto*
note *abscissa* = *le-less-trans*[*OF abs-conv-abscissa-completely-multiplicative-log-deriv*[*OF* $*$]]
have $(\lambda p.\ ln\ (real\ p) * (1\ /\ (1 - fds\text{-}nth\ fds\text{-}zeta\ p\ /\ p\ powr\ s) - 1))$
$abs\text{-}summable\text{-}on\ \{p.\ prime\ p\}$
using *eval-fds-logderiv-completely-multiplicative*[*OF* $*$, *of s*] *assms* **by** *auto*
also have *?this* $\longleftrightarrow$ $(\lambda p.\ ln\ (real\ p)\ /\ (p\ powr\ s - 1))$ *abs-summable-on* {*p. prime p*} **using** *assms*
by (*intro abs-summable-on-cong*) (*auto simp*: *fds-nth-zeta divide-simps dest*: *prime-gt-1-nat*)
finally show *?th1* **.**

from *assms* **have** *ev*: *eventually* $(\lambda z.\ z \in \{z.\ Re\ z > 1\})$ (*nhds s*)
by (*intro eventually-nhds-in-open open-halfspace-Re-gt*) *auto*
have *deriv zeta s* = *deriv* (*eval-fds fds-zeta*) *s*
by (*intro deriv-cong-ev*[*OF eventually-mono*[*OF ev*]]) (*auto simp*: *eval-fds-zeta*)
also have *deriv* (*eval-fds fds-zeta*) *s* / *zeta s* = *eval-fds* (*fds-deriv fds-zeta* / *fds-zeta*) *s*
using *assms zeta-Re-gt-1-nonzero*[*of s*]
by (*subst eval-fds-log-deriv*) (*auto simp*: *eval-fds-zeta eval-fds-deriv intro*!: *abscissa*)
also have *eval-fds* (*fds-deriv fds-zeta* / *fds-zeta*) *s* =
$-(\sum_a p \mid prime\ p.\ ln\ (real\ p) * (1\ /\ (1 - fds\text{-}nth\ fds\text{-}zeta\ p\ /\ p\ powr\ s) - 1))$
(**is** - = $-$*?S*) **using** *eval-fds-logderiv-completely-multiplicative*[*OF* $*$, *of s*] *assms*
by *auto*
also have *?S* = $(\sum_a p \mid prime\ p.\ ln\ (real\ p)\ /\ (p\ powr\ s - 1))$ **using** *assms*
by (*intro infsetsum-cong*) (*auto simp*: *fds-nth-zeta divide-simps dest*: *prime-gt-1-nat*)
finally show *?th2* **.**
qed

lemma *sums-logderiv-zeta*:
assumes *Re s* > 1
shows $(\lambda p.\ if\ prime\ p\ then\ of\text{-}real\ (ln\ (real\ p))\ /\ (of\text{-}nat\ p\ powr\ s - 1)\ else\ 0)$ *sums*
$-(deriv\ zeta\ s\ /\ zeta\ s)$ (**is** *?f sums* -)
proof −
note $*$ = *eval-fds-logderiv-zeta*[*OF assms*]
from *sums-infsetsum-nat*[*OF* $*(1)$] **and** $*(2)$ **show** *?thesis* **by** *simp*
qed

lemma *range-add-nat*: *range* $(\lambda n.\ n + c) = \{(c::nat)..\}$
using *Nat.le-imp-diff-is-add* **by** *auto*

lemma *abs-summable-hurwitz-zeta*:
assumes *Re s > 1 a + real b > 0*
shows *(λn. 1 / (of-nat n + a) powr s) abs-summable-on {b..}*
proof −
from *assms* **have** *summable (λn. cmod (1 / (of-nat (n + b) + a) powr s))*
using *summable-hurwitz-zeta-real[of Re s a + b]*
by (*auto simp*: *norm-divide powr-minus field-simps norm-powr-real-powr*)
hence *(λn. 1 / (of-nat (n + b) + a) powr s) abs-summable-on UNIV*
by (*auto simp*: *abs-summable-on-nat-iff' add-ac*)
also have *?this ⟷ (λn. 1 / (of-nat n + a) powr s) abs-summable-on range (λn. n + b)*
by (*rule abs-summable-on-reindex-iff*) *auto*
also have *range (λn. n + b) = {b..}* **by** (*rule range-add-nat*)
finally show *?thesis* **.**
qed

lemma *hurwitz-zeta-nat-conv-infsetsum*:
assumes *a > 0* **and** *Re s > 1*
shows *hurwitz-zeta (real a) s = (∑ a n. of-nat (n + a) powr −s)*
hurwitz-zeta (real a) s = (∑ a n∈{a..}. of-nat n powr −s)
proof −
have *hurwitz-zeta (real a) s = (∑ n. of-nat (n + a) powr −s)*
using *assms* **by** (*subst hurwitz-zeta-conv-suminf*) *auto*
also have *... = (∑ a n. of-nat (n + a) powr −s)*
using *abs-summable-hurwitz-zeta[of s a 0] assms*
by (*intro infsetsum-nat' [symmetric]*) (*auto simp*: *powr-minus field-simps*)
finally show *hurwitz-zeta (real a) s = (∑ a n. of-nat (n + a) powr −s)* **.**
also have *... = (∑ a n∈range (λn. n + a). of-nat n powr −s)*
by (*rule infsetsum-reindex [symmetric]*) *auto*
also have *range (λn. n + a) = {a..}* **by** (*rule range-add-nat*)
finally show *hurwitz-zeta (real a) s = (∑ a n∈{a..}. of-nat n powr −s)* **.**
qed

lemma *pre-zeta-bound*:
assumes *0 < Re s* **and** *a*: *a > 0*
shows *norm (pre-zeta a s) ≤ (1 + norm s / Re s) / 2 * a powr −Re s*
proof −
let *?f = λx. − (s * (x + a) powr (−1−s))*
let *?g' = λx. norm s * (x + a) powr (−1−Re s)*
let *?g = λx. −norm s / Re s * (x + a) powr (−Re s)*
define *R* **where** *R = EM-remainder 1 ?f 0*
have [*simp*]: *−Re s − 1 = −1 − Re s* **by** (*simp add*: *algebra-simps*)

have *|frac x − 1 / 2| ≤ 1 / 2* **for** *x* :: *real* **unfolding** *frac-def*
by *linarith*
hence *|pbernpoly (Suc 0) x| ≤ 1 / 2* **for** *x*
by (*simp add*: *pbernpoly-def bernpoly-def*)
moreover have *((λb. cmod s * (b + a) powr − Re s / Re s) ⟶ 0) at-top*

```
    using ‹Re s > 0› ‹a > 0› by real-asymp
  ultimately have ∗: ∀ x. x ≥ real 0 ⟶ norm (EM-remainder 1 ?f (int x)) ≤
                       (1 / 2) / fact 1 ∗ (−?g (real x))
    using ‹a > 0› ‹Re s > 0›
    by (intro norm-EM-remainder-le-strong-nat'[where g' = ?g' and Y = {}])
       (auto intro!: continuous-intros derivative-eq-intros
             simp: field-simps norm-mult norm-powr-real-powr add-eq-0-iff)
  have R: norm R ≤ norm s / (2 ∗ Re s) ∗ a powr −Re s
    unfolding R-def using spec[OF ∗, of 0] by simp

  from assms have pre-zeta a s = a powr −s / 2 + R
    by (simp add: pre-zeta-def pre-zeta-aux-def R-def)
  also have norm ... ≤ a powr −Re s / 2 + norm s / (2 ∗ Re s) ∗ a powr −Re
s using a
   by (intro order.trans[OF norm-triangle-ineq] add-mono R) (auto simp: norm-powr-real-powr)
  also have ... = (1 + norm s / Re s) / 2 ∗ a powr −Re s
    by (simp add: field-simps)
  finally show ?thesis .
qed

lemma pre-zeta-bound':
  assumes 0 < Re s and a: a > 0
  shows   norm (pre-zeta a s) ≤ norm s / (Re s ∗ a powr Re s)
proof −
  from assms have norm (pre-zeta a s) ≤ (1 + norm s / Re s) / 2 ∗ a powr −Re
s
    by (intro pre-zeta-bound) auto
  also have ... = (Re s + norm s) / 2 / (Re s ∗ a powr Re s)
    using assms by (auto simp: field-simps powr-minus)
  also have Re s + norm s ≤ norm s + norm s by (intro add-right-mono com-
plex-Re-le-cmod)
  also have (norm s + norm s) / 2 = norm s by simp
  finally show norm (pre-zeta a s) ≤ norm s / (Re s ∗ a powr Re s)
    using assms by (simp add: divide-right-mono)
qed

lemma deriv-zeta-eq:
  assumes s: s ≠ 1
  shows   deriv zeta s = deriv (pre-zeta 1) s − 1 / (s − 1)^2
proof −
  from s have ev: eventually (λz. z ≠ 1) (nhds s) by (intro t1-space-nhds)
  have [derivative-intros]: (pre-zeta 1 has-field-derivative deriv (pre-zeta 1) s) (at
s)
    by (intro holomorphic-derivI[of - UNIV] holomorphic-intros) auto
  have ((λs. pre-zeta 1 s + 1 / (s − 1)) has-field-derivative
        (deriv (pre-zeta 1) s − 1 / (s − 1)^2)) (at s)
    using s by (auto intro!: derivative-eq-intros simp: power2-eq-square)
  also have ?this ⟷ (zeta has-field-derivative (deriv (pre-zeta 1) s − 1 / (s −
1)^2)) (at s)
```

```
    by (intro has-field-derivative-cong-ev eventually-mono[OF ev])
       (auto simp: zeta-def hurwitz-zeta-def)
  finally show ?thesis by (rule DERIV-imp-deriv)
qed

lemma zeta-remove-zero:
  assumes Re s ≥ 1
  shows   (s − 1) ∗ pre-zeta 1 s + 1 ≠ 0
proof (cases s = 1)
  case False
  hence (s − 1) ∗ pre-zeta 1 s + 1 = (s − 1) ∗ zeta s
    by (simp add: zeta-def hurwitz-zeta-def divide-simps)
  also from False assms have ... ≠ 0 using zeta-Re-ge-1-nonzero[of s] by auto
  finally show ?thesis .
qed auto

lemma eval-fds-deriv-zeta:
  assumes Re s > 1
  shows   eval-fds (fds-deriv fds-zeta) s = deriv zeta s
proof −
  have ev: eventually (λz. z ∈ {z. Re z > 1}) (nhds s)
    using assms by (intro eventually-nhds-in-open open-halfspace-Re-gt) auto
  from assms have eval-fds (fds-deriv fds-zeta) s = deriv (eval-fds fds-zeta) s
    by (subst eval-fds-deriv) auto
  also have ... = deriv zeta s
    by (intro deriv-cong-ev eventually-mono[OF ev]) (auto simp: eval-fds-zeta)
  finally show ?thesis .
qed

lemma le-nat-iff': x ≤ nat y ⟷ x = 0 ∧ y ≤ 0 ∨ int x ≤ y
  by auto

lemma sum-upto-plus1:
  assumes x ≥ 0
  shows   sum-upto f (x + 1) = sum-upto f x + f (Suc (nat ⌊x⌋))
proof −
  have sum-upto f (x + 1) = sum f {0<..Suc (nat ⌊x⌋)}
    using assms by (simp add: sum-upto-altdef nat-add-distrib)
  also have {0<..Suc (nat ⌊x⌋)} = insert (Suc (nat ⌊x⌋)) {0<..nat ⌊x⌋}
    by auto
  also have sum f ... = sum-upto f x + f (Suc (nat ⌊x⌋))
    by (subst sum.insert) (auto simp: sum-upto-altdef add-ac)
  finally show ?thesis .
qed

lemma sum-upto-minus1:
  assumes x ≥ 1
  shows   sum-upto f (x − 1) = (sum-upto f x − f (nat ⌊x⌋) :: 'a :: ab-group-add)
  using sum-upto-plus1[of x − 1 f] assms by (simp add: algebra-simps nat-diff-distrib)
```

lemma *integral-smallo*:
fixes $f\ g\ g' :: real \Rightarrow real$
assumes $f \in o(g')$ **and** *filterlim g at-top at-top*
assumes $\bigwedge a'\ x.\ a \le a' \Longrightarrow a' \le x \Longrightarrow f$ *integrable-on* $\{a'..x\}$
assumes *deriv*: $\bigwedge x.\ x \ge a \Longrightarrow (g$ *has-field-derivative* $g'\ x)$ *(at x)*
assumes *cont*: *continuous-on* $\{a..\}\ g'$
assumes *nonneg*: $\bigwedge x.\ x \ge a \Longrightarrow g'\ x \ge 0$
shows $(\lambda x.$ *integral* $\{a..x\}\ f) \in o(g)$
proof (*rule landau-o.smallI*)
fix $c :: real$ **assume** *c*: $c > 0$
note [*continuous-intros*] = *continuous-on-subset*[*OF cont*]
define c' **where** $c' = c\ /\ 2$
from *c* **have** c': $c' > 0$ **by** (*simp add*: c'-*def*)
from *landau-o.smallD*[*OF assms*(*1*) *this*]
obtain *b* **where** *b*: $\bigwedge x.\ x \ge b \Longrightarrow norm\ (f\ x) \le c' * norm\ (g'\ x)$
unfolding *eventually-at-top-linorder* **by** *blast*
define b' **where** $b' = max\ a\ b$
define *D* **where** $D = norm\ (integral\ \{a..b'\}\ f)$

have *filterlim* $(\lambda x.\ c' * g\ x)$ *at-top at-top*
using c' **by** (*intro filterlim-tendsto-pos-mult-at-top*[*OF tendsto-const*] *assms*)
hence *eventually* $(\lambda x.\ c' * g\ x \ge D - c' * g\ b')$ *at-top*
by (*auto simp*: *filterlim-at-top*)
thus *eventually* $(\lambda x.\ norm\ (integral\ \{a..x\}\ f) \le c * norm\ (g\ x))$ *at-top*
using *eventually-ge-at-top*[*of* b']
proof *eventually-elim*
case (*elim x*)
have b': $a \le b'\ b \le b'$ **by** (*auto simp*: b'-*def*)
from *elim* b' **have** *integrable*: $(\lambda x.\ |g'\ x|)$ *integrable-on* $\{b'..x\}$
by (*intro integrable-continuous-real continuous-intros*) *auto*
have *integral* $\{a..x\}\ f =$ *integral* $\{a..b'\}\ f +$ *integral* $\{b'..x\}\ f$
using *elim* b' **by** (*intro Henstock-Kurzweil-Integration.integral-combine* [*symmetric*]
assms) *auto*
also have $norm\ \ldots \le D + norm\ (integral\ \{b'..x\}\ f)$
unfolding *D-def* **by** (*rule norm-triangle-ineq*)
also have $norm\ (integral\ \{b'..x\}\ f) \le integral\ \{b'..x\}\ (\lambda x.\ c' * norm\ (g'\ x))$
using b' *elim assms* c' *integrable* **by** (*intro integral-norm-bound-integral b*
assms) *auto*
also have $\ldots = c' *$ *integral* $\{b'..x\}\ (\lambda x.\ |g'\ x|)$ **by** *simp*
also have *integral* $\{b'..x\}\ (\lambda x.\ |g'\ x|) =$ *integral* $\{b'..x\}\ g'$
using *assms* b' **by** (*intro integral-cong*) *auto*
also have $(g'$ *has-integral* $(g\ x - g\ b'))\ \{b'..x\}$ **using** b' *elim*
by (*intro fundamental-theorem-of-calculus*)
(*auto simp flip*: *has-real-derivative-iff-has-vector-derivative*
intro!: *has-field-derivative-at-within*[*OF deriv*])
hence *integral* $\{b'..x\}\ g' = g\ x - g\ b'$
by (*simp add*: *has-integral-iff*)
also have $D + c' * (g\ x - g\ b') \le c * g\ x$

```
    using elim by (simp add: field-simps c'-def)
  also have ... ≤ c * norm (g x)
    using c by (intro mult-left-mono) auto
  finally show ?case by simp
 qed
qed

lemma integral-bigo:
 fixes f g g' :: real ⇒ real
 assumes f ∈ O(g') and filterlim g at-top at-top
 assumes ⋀a' x. a ≤ a' ⟹ a' ≤ x ⟹ f integrable-on {a'..x}
 assumes deriv: ⋀x. x ≥ a ⟹ (g has-field-derivative g' x) (at x within {a..})
 assumes cont: continuous-on {a..} g'
 assumes nonneg: ⋀x. x ≥ a ⟹ g' x ≥ 0
 shows   (λx. integral {a..x} f) ∈ O(g)
proof −
 note [continuous-intros] = continuous-on-subset[OF cont]
 from landau-o.bigE[OF assms(1)]
  obtain c b where c: c > 0 and b: ⋀x. x ≥ b ⟹ norm (f x) ≤ c * norm (g'
x)
    unfolding eventually-at-top-linorder by metis
 define c' where c' = c / 2
 define b' where b' = max a b
 define D where D = norm (integral {a..b'} f)

 have filterlim (λx. c * g x) at-top at-top
  using c by (intro filterlim-tendsto-pos-mult-at-top[OF tendsto-const] assms)
 hence eventually (λx. c * g x ≥ D − c * g b') at-top
  by (auto simp: filterlim-at-top)
 hence eventually (λx. norm (integral {a..x} f) ≤ 2 * c * norm (g x)) at-top
  using eventually-ge-at-top[of b']
 proof eventually-elim
  case (elim x)
  have b': a ≤ b' b ≤ b' by (auto simp: b'-def)
  from elim b' have integrable: (λx. |g' x|) integrable-on {b'..x}
   by (intro integrable-continuous-real continuous-intros) auto
  have integral {a..x} f = integral {a..b'} f + integral {b'..x} f
   using elim b' by (intro Henstock-Kurzweil-Integration.integral-combine [symmetric]
assms) auto
  also have norm ... ≤ D + norm (integral {b'..x} f)
   unfolding D-def by (rule norm-triangle-ineq)
  also have norm (integral {b'..x} f) ≤ integral {b'..x} (λx. c * norm (g' x))
    using b' elim assms c integrable by (intro integral-norm-bound-integral b
assms) auto
  also have ... = c * integral {b'..x} (λx. |g' x|) by simp
  also have integral {b'..x} (λx. |g' x|) = integral {b'..x} g'
   using assms b' by (intro integral-cong) auto
  also have (g' has-integral (g x − g b')) {b'..x} using b' elim
   by (intro fundamental-theorem-of-calculus)
```

```
      (auto simp flip: has-real-derivative-iff-has-vector-derivative
          intro!: DERIV-subset[OF deriv])
    hence integral {b'..x} g' = g x - g b'
      by (simp add: has-integral-iff)
    also have D + c * (g x - g b') ≤ 2 * c * g x
      using elim by (simp add: field-simps c'-def)
    also have ... ≤ 2 * c * norm (g x)
      using c by (intro mult-left-mono) auto
    finally show ?case by simp
  qed
  thus ?thesis by (rule bigoI)
qed

lemma primepows-le-subset:
  assumes x: x > 0 and l: l > 0
  shows  {(p, i). prime p ∧ l ≤ i ∧ real (p ^ i) ≤ x} ⊆ {..nat ⌊root l x⌋} × {..nat
⌊log 2 x⌋}
proof safe
  fix p i :: nat assume pi: prime p i ≥ l real (p ^ i) ≤ x
  have real p ^ l ≤ real p ^ i using pi x l
    by (intro power-increasing) (auto dest: prime-gt-0-nat)
  also have ... ≤ x using pi by simp
  finally have root l (real p ^ l) ≤ root l x
    using x pi l by (subst real-root-le-iff) auto
  also have root l (real p ^ l) = real p
    using pi l by (subst real-root-pos2) auto
  finally show p ≤ nat ⌊root l x⌋ using pi l x by (simp add: le-nat-iff' le-floor-iff)

  from pi have 2 ^ i ≤ real p ^ i using l
    by (intro power-mono) (auto dest: prime-gt-1-nat)
  also have ... ≤ x using pi by simp
  finally show i ≤ nat ⌊log 2 x⌋ using pi x
    by (auto simp: le-nat-iff' le-floor-iff le-log-iff powr-realpow)
qed

lemma mangoldt-non-primepow: ¬primepow n ⟹ mangoldt n = 0
  by (auto simp: mangoldt-def)


lemma ln-minus-ln-floor-bigo: (λx. ln x - ln (real (nat ⌊x⌋))) ∈ O(λ-. 1)
proof (intro le-imp-bigo-real[of 1] eventually-mono[OF eventually-ge-at-top[of 1]])
  fix x :: real assume x: x ≥ 1
  from x have *: x - real (nat ⌊x⌋) ≤ 1 by linarith
  from x have ln x - ln (real (nat ⌊x⌋)) ≤ (x - real (nat ⌊x⌋)) / real (nat ⌊x⌋)
    by (intro ln-diff-le) auto
  also have ... ≤ 1 / 1 using x * by (intro frac-le) auto
  finally show ln x - ln (real (nat ⌊x⌋)) ≤ 1 * 1 by simp
qed auto
```

lemma *cos-geD*:
assumes *cos x* ≥ *cos a* 0 ≤ *a a* ≤ *pi* −*pi* ≤ *x x* ≤ *pi*
shows *x* ∈ {−*a*..*a*}
proof (*cases x* ≥ 0)
case *True*
with *assms* **show** *?thesis*
by (*subst* (*asm*) *cos-mono-le-eq*) *auto*
next
case *False*
with *assms* **show** *?thesis* **using** *cos-mono-le-eq*[*of a* −*x*]
by *auto*
qed

lemma *path-image-part-circlepath-same-Re*:
assumes 0 ≤ *b b* ≤ *pi a* = −*b r* ≥ 0
shows *path-image* (*part-circlepath c r a b*) = *sphere c r* ∩ {*s*. *Re s* ≥ *Re c* + *r* ∗ *cos a*}
proof *safe*
fix *z* **assume** *z* ∈ *path-image* (*part-circlepath c r a b*)
with *assms* **obtain** *t* **where** *t*: *t* ∈ {*a*..*b*} *z* = *c* + *of-real r* ∗ *cis t*
by (*auto simp*: *path-image-part-circlepath exp-eq-polar*)
from *t* **and** *assms* **show** *z* ∈ *sphere c r*
by (*auto simp*: *dist-norm norm-mult*)
from *t* **and** *assms* **show** *Re z* ≥ *Re c* + *r* ∗ *cos a*
using *cos-monotone-0-pi-le*[*of t b*] *cos-monotone-minus-pi-0'*[*of a t*]
by (*cases t* ≥ 0) (*auto intro*!: *mult-left-mono*)
next
fix *z* **assume** *z*: *z* ∈ *sphere c r Re z* ≥ *Re c* + *r* ∗ *cos a*
show *z* ∈ *path-image* (*part-circlepath c r a b*)
proof (*cases r* = 0)
case *False*
with *assms* **have** *r*: *r* > 0 **by** *simp*
with *z* **have** *z-eq*: *z* = *c* + *r* ∗ *cis* (*Arg* (*z* − *c*))
using *Arg-eq*[*of z* − *c*] **by** (*auto simp*: *dist-norm exp-eq-polar norm-minus-commute*)
moreover from *z*(2) *r assms* **have** *cos b* ≤ *cos* (*Arg* (*z* − *c*))
by (*subst* (*asm*) *z-eq*) *auto*
with *assms* **have** *Arg* (*z* − *c*) ∈ {−*b*..*b*}
using *Arg-le-pi*[*of z* − *c*] *mpi-less-Arg*[*of z* − *c*] **by** (*intro cos-geD*) *auto*
ultimately show *z* ∈ *path-image* (*part-circlepath c r a b*)
using *assms* **by** (*subst path-image-part-circlepath*) (*auto simp*: *exp-eq-polar*)
qed (*insert assms z*, *auto simp*: *path-image-part-circlepath*)
qed

lemma *part-circlepath-rotate-left*:
part-circlepath c r (*x* + *a*) (*x* + *b*) = (λ*z*. *c* + *cis x* ∗ (*z* − *c*)) ∘ *part-circlepath c r a b*
by (*simp add*: *part-circlepath-def exp-eq-polar fun-eq-iff*
linepath-translate-left linepath-translate-right cis-mult add-ac)

lemma *part-circlepath-rotate-right*:
part-circlepath c r (a + x) (b + x) = (λz. c + cis x ∗ (z − c)) ∘ part-circlepath c r a b
by (*simp add*: *part-circlepath-def exp-eq-polar fun-eq-iff linepath-translate-left linepath-translate-right cis-mult add-ac*)

lemma *path-image-semicircle-Re-ge*:
assumes *r ≥ 0*
shows *path-image (part-circlepath c r (−pi/2) (pi/2)) = sphere c r ∩ {s. Re s ≥ Re c}*
by (*subst path-image-part-circlepath-same-Re*) (*simp-all add*: *assms*)

lemma *sphere-rotate*: *(λz. c + cis x ∗ (z − c)) ‘ sphere c r = sphere c r*
proof *safe*
fix *z* **assume** *z*: *z ∈ sphere c r*
hence *z = c + cis x ∗ (c + cis (−x) ∗ (z − c) − c)*
c + cis (−x) ∗ (z − c) ∈ sphere c r
by (*auto simp*: *dist-norm norm-mult norm-minus-commute cis-conv-exp exp-minus field-simps norm-divide*)
with *z* **show** *z ∈ (λz. c + cis x ∗ (z − c)) ‘ sphere c r* **by** *blast*
qed (*auto simp*: *dist-norm norm-minus-commute norm-mult*)

lemma *path-image-semicircle-Re-le*:
assumes *r ≥ 0*
shows *path-image (part-circlepath c r (pi/2) (3/2∗pi)) = sphere c r ∩ {s. Re s ≤ Re c}*
proof −
let *?f = (λz. c + cis pi ∗ (z − c))*
have ∗: *part-circlepath c r (pi/2) (3/2∗pi) = part-circlepath c r (pi + (−pi/2)) (pi + pi/2)*
by *simp*
have *path-image (part-circlepath c r (pi/2) (3/2∗pi)) = ?f ‘ sphere c r ∩ ?f ‘ {s. Re c ≤ Re s}*
unfolding ∗ *part-circlepath-rotate-left path-image-compose path-image-semicircle-Re-ge*[*OF assms*]
by *auto*
also have *?f ‘ sphere c r = sphere c r*
by (*rule sphere-rotate*)
also have *?f ‘ {s. Re c ≤ Re s} = {s. Re c ≥ Re s}*
by (*auto simp*: *image-iff intro*!: *exI*[*of - 2 ∗ c − x* **for** *x*])
finally show *?thesis* **.**
qed

lemma *path-image-semicircle-Im-ge*:
assumes *r ≥ 0*
shows *path-image (part-circlepath c r 0 pi) = sphere c r ∩ {s. Im s ≥ Im c}*

proof −
let *?f* = (λ*z*. *c* + *cis* (*pi*/2) ∗ (*z* − *c*))
have ∗: *part-circlepath c r 0 pi* = *part-circlepath c r* (*pi* / 2 + (−*pi*/2)) (*pi* / 2 + *pi*/2)
by *simp*
have *path-image* (*part-circlepath c r 0 pi*) =
?f ‘ *sphere c r* ∩ *?f* ‘ {*s*. *Re c* ≤ *Re s*}
unfolding ∗ *part-circlepath-rotate-left path-image-compose path-image-semicircle-Re-ge*[*OF assms*]
by *auto*
also have *?f* ‘ *sphere c r* = *sphere c r*
by (*rule sphere-rotate*)
also have *?f* ‘ {*s*. *Re c* ≤ *Re s*} = {*s*. *Im c* ≤ *Im s*}
by (*auto simp*: *image-iff intro*!: *exI*[*of* - *c* − i ∗ (*x* − *c*) **for** *x*])
finally show *?thesis* **.**
qed

lemma *path-image-semicircle-Im-le*:
assumes *r* ≥ *0*
shows *path-image* (*part-circlepath c r pi* (*2* ∗ *pi*)) =
sphere c r ∩ {*s*. *Im s* ≤ *Im c*}
proof −
let *?f* = (λ*z*. *c* + *cis* (*3*∗*pi*/*2*) ∗ (*z* − *c*))
have ∗: *part-circlepath c r pi* (*2*∗*pi*) = *part-circlepath c r* (*3*∗*pi*/*2* + (−*pi*/*2*)) (*3*∗*pi*/*2* + *pi*/*2*)
by *simp*
have *path-image* (*part-circlepath c r pi* (*2* ∗ *pi*)) =
?f ‘ *sphere c r* ∩ *?f* ‘ {*s*. *Re c* ≤ *Re s*}
unfolding ∗ *part-circlepath-rotate-left path-image-compose path-image-semicircle-Re-ge*[*OF assms*]
by *auto*
also have *?f* ‘ *sphere c r* = *sphere c r*
by (*rule sphere-rotate*)
also have *cis* (*3* ∗ *pi* / *2*) = −i
using *cis-mult*[*of pi pi* / *2*] **by** *simp*
hence *?f* ‘ {*s*. *Re c* ≤ *Re s*} = {*s*. *Im c* ≥ *Im s*}
by (*auto simp*: *image-iff intro*!: *exI*[*of* - *c* + i ∗ (*x* − *c*) **for** *x*])
finally show *?thesis* **.**
qed

lemma *eval-fds-logderiv-zeta-real*:
assumes *x* > (*1* :: *real*)
shows (λ*p*. *ln* (*real p*) / (*p powr x* − *1*)) *abs-summable-on* {*p*. *prime p*} (**is** *?th1*)
and *deriv zeta* (*of-real x*) / *zeta* (*of-real x*) =
−*of-real* (∑ *a p* | *prime p*. *ln* (*real p*) / (*p powr x* − *1*)) (**is** *?th2*)
proof −
have (λ*p*. *Re* (*of-real* (*ln* (*real p*)) / (*of-nat p powr of-real x* − *1*)))
abs-summable-on {*p*. *prime p*} **using** *assms*

by (*intro abs-summable-Re eval-fds-logderiv-zeta*) *auto*
also have *?this* $\longleftrightarrow$ *?th1*
by (*intro abs-summable-on-cong*) (*auto simp*: *powr-Reals-eq*)
finally show *?th1* **.**
show *?th2* **using** *assms*
by (*subst eval-fds-logderiv-zeta*) (*auto simp*: *infsetsum-of-real* [*symmetric*] *powr-Reals-eq*)
qed

lemma
fixes *a b c d* :: *real*
assumes *ab*: $d * a + b \geq 1$ **and** *c*: $c < -1$ **and** *d*: $d > 0$
defines $C \equiv -((ln\ (d * a + b) - 1\ /\ (c + 1)) * (d * a + b)\ powr\ (c + 1)\ /\ (d * (c + 1)))$
shows *set-integrable-ln-powr-at-top*:
$(\lambda x.\ (ln\ (d * x + b) * ((d * x + b)\ powr\ c)))$ *absolutely-integrable-on* $\{a<..\}$ (**is** *?th1*)
and *set-lebesgue-integral-ln-powr-at-top*:
$(\int x{\in}\{a<..\}.\ (ln\ (d * x + b) * ((d * x + b)\ powr\ c))\ \partial lborel) = C$ (**is** *?th2*)
and *ln-powr-has-integral-at-top*:
$((\lambda x.\ ln\ (d * x + b) * (d * x + b)\ powr\ c)\ has\text{-}integral\ C)\ \{a<..\}$ (**is** *?th3*)
proof −
define *f* **where** $f = (\lambda x.\ ln\ (d * x + b) * (d * x + b)\ powr\ c)$
define *F* **where** $F = (\lambda x.\ (ln\ (d * x + b) - 1\ /\ (c + 1)) * (d * x + b)\ powr\ (c + 1)\ /\ (d * (c + 1)))$

have $*$: (*F has-field-derivative f x*) (*at x*) *isCont f x f x* ≥ 0 **if** $x > a$ **for** *x*
proof −
have $1 \leq d * a + b$ **by** *fact*
also have $\ldots < d * x + b$ **using** *that assms*
by (*intro add-strict-right-mono mult-strict-left-mono*)
finally have *gt-1*: $d * x + b > 1$ **.**
show (*F has-field-derivative f x*) (*at x*) *isCont f x* **using** *ab c d gt-1*
by (*auto simp*: *F-def f-def divide-simps intro*!: *derivative-eq-intros continuous-intros*)
(*auto simp*: *algebra-simps powr-add*)*?*
show $f\ x \geq 0$ **using** *gt-1* **by** (*auto simp*: *f-def*)
qed

have *limits*: $((F \circ real\text{-}of\text{-}ereal) \longrightarrow F\ a)\ (at\text{-}right\ (ereal\ a))$
$((F \circ real\text{-}of\text{-}ereal) \longrightarrow 0)\ (at\text{-}left\ \infty)$
using *c ab d* **unfolding** *ereal-tendsto-simps1 F-def* **by** (*real-asymp*; *simp add*: *field-simps*)+
have *1*: *set-integrable lborel* (*einterval a* ∞) *f* **using** *ab c limits*
by (*intro interval-integral-FTC-nonneg*) (*auto intro*!: $*$ *AE-I2*)
thus *2*: *f absolutely-integrable-on* $\{a<..\}$
by (*auto simp*: *set-integrable-def integrable-completion*)
have (*LBINT x=ereal a..*∞. *f x*) $= 0 - F\ a$ **using** *ab c limits*
by (*intro interval-integral-FTC-nonneg*) (*auto intro*!: $*$)

```
  thus 3: ?th2
    by (simp add: interval-integral-to-infinity-eq F-def f-def C-def)
  show ?th3
    using set-borel-integral-eq-integral[OF 1] 3 by (simp add: has-integral-iff f-def
C-def)
qed

lemma ln-fact-conv-sum-upto: ln (fact n) = sum-upto ln n
  by (induction n) (auto simp: sum-upto-plus1 add.commute[of 1] ln-mult)

lemma sum-upto-ln-conv-ln-fact: sum-upto ln x = ln (fact (nat ⌊x⌋))
  by (simp add: ln-fact-conv-sum-upto sum-upto-altdef)

lemma real-of-nat-div: real (a div b) = real-of-int ⌊real a / real b⌋
  by (simp add: floor-divide-of-nat-eq)

lemma measurable-sum-upto [measurable]:
  fixes f :: 'a ⇒ nat ⇒ real
  assumes [measurable]: ⋀y. (λt. f t y) ∈ M →M borel
  assumes [measurable]: x ∈ M →M borel
  shows (λt. sum-upto (f t) (x t)) ∈ M →M borel
proof −
  have meas: (λt. set-lebesgue-integral lborel {y. y ≥ 0 ∧ y − real (nat ⌊x t⌋) ≤
0} (λy. f t (nat ⌈y⌉)))
        ∈ M →M borel (is ?f ∈ -) unfolding set-lebesgue-integral-def
    by measurable
  also have ?f = (λt. sum-upto (f t) (x t))
  proof
    fix t :: 'a
    show ?f t = sum-upto (f t) (x t)
    proof (cases x t < 1)
      case True
      hence {y. y ≥ 0 ∧ y − real (nat ⌊x t⌋) ≤ 0} = {0} by auto
      thus ?thesis using True
        by (simp add: set-integral-at-point sum-upto-altdef)
    next
      case False
      define n where n = nat ⌊x t⌋
      from False have n > 0 by (auto simp: n-def)

      have ∗: ((λx. f t (nat ⌈x⌉)) has-integral sum (f t) {0<..n}) {real 0..real n}
        using ‹n > 0› by (intro nat-sum-has-integral-ceiling) auto

      have ∗∗: (λx. f t (nat ⌈x⌉)) absolutely-integrable-on {real 0..real n}
      proof (rule absolutely-integrable-absolutely-integrable-ubound)
        show (λ-. MAX n∈{0..n}. |f t n|) absolutely-integrable-on {real 0..real n}
          using ‹n > 0› by (subst absolutely-integrable-on-iff-nonneg)
                           (auto simp: Max-ge-iff intro!: exI[of - f t 0])
        show (λx. f t (nat ⌈x⌉)) integrable-on {real 0..real n}
```

```
      using ∗ by (simp add: has-integral-iff)
    next
      fix y :: real assume y: y ∈ {real 0..real n}
      have f t (nat ⌈y⌉) ≤ |f t (nat ⌈y⌉)|
        by simp
      also have ... ≤ (MAX n∈{0..n}. |f t n|)
        using y by (intro Max.coboundedI) auto
      finally show f t (nat ⌈y⌉) ≤ (MAX n∈{0..n}. |f t n|) .
    qed
    have sum (f t) {0<..n} = (∫ x∈{real 0..real n}. f t (nat ⌈x⌉) ∂lebesgue)
      using has-integral-set-lebesgue[OF **] ∗ by (simp add: has-integral-iff)
    also have ... = (∫ x∈{real 0..real n}. f t (nat ⌈x⌉) ∂lborel)
      unfolding set-lebesgue-integral-def by (subst integral-completion) auto
    also have {real 0..real n} = {y. 0 ≤ y ∧ y − real (nat ⌊x t⌋) ≤ 0}
      by (auto simp: n-def)
    also have sum (f t) {0<..n} = sum-upto (f t) (x t)
      by (simp add: sum-upto-altdef n-def)
    finally show ?thesis ..
  qed
 qed
 finally show ?thesis .
qed

end
```

2 Ingham's Tauberian Theorem

```
theory Newman-Ingham-Tauberian
imports
  HOL−Real-Asymp.Real-Asymp
  Prime-Number-Theorem-Library
begin
```

In his proof of the Prime Number Theorem, Newman [6] uses a Tauberian theorem that was first proven by Ingham. Newman gives a nice and straightforward proof of this theorem based on contour integration. This section will be concerned with proving this theorem.

This Tauberian theorem is probably the part of the Newman's proof of the Prime Number Theorem where most of the "heavy lifting" is done. Its purpose is to extend the summability of a Dirichlet series with bounded coefficients from the region $\Re(s) > 1$ to $\Re(s) \geq 1$.

In order to show it, we first require a number of auxiliary bounding lemmas.

```
lemma newman-ingham-aux1:
  fixes R :: real and z :: complex
  assumes R: R > 0 and z : norm z = R
  shows   norm (1 / z + z / R^2) = 2 ∗ |Re z| / R^2
proof −
```

```
  from z and R have [simp]: z ≠ 0 by auto
  have 1 / z + z / R² = (R² + z²) * (1 / R² / z) using R
    by (simp add: field-simps power2-eq-square)
  also have norm ... = norm (R² + z²) / R ^ 3
    by (simp add: numeral-3-eq-3 z norm-divide norm-mult power2-eq-square)
  also have R² + z² = z * (z + cnj z) using complex-norm-square[of z]
    by (simp add: z power2-eq-square algebra-simps)
  also have norm ... = 2 * |Re z| * R
    by (subst complex-add-cnj) (simp-all add: z norm-mult)
  also have ... / R ^ 3 = 2 * |Re z| / R²
    using R by (simp add: field-simps numeral-3-eq-3 power2-eq-square)
  finally show ?thesis .
qed

lemma newman-ingham-aux2:
  fixes m :: nat and w z :: complex
  assumes 1 ≤ m 1 ≤ Re w 0 < Re z and f: ⋀n. 1 ≤ n ⟹ norm (f n) ≤ C
  shows norm (∑ n=1..m. f n / n powr (w − z)) ≤ C * (m powr Re z) * (1 / m
+ 1 / Re z)
proof −
  have [simp]: C ≥ 0 by (rule order.trans[OF - f[of 1]]) auto
  have norm (∑ n=1..m. f n / n powr (w − z)) ≤ (∑ n=1..m. C / n powr (1 −
Re z))
    by (rule sum-norm-le)
       (insert assms, auto simp: norm-divide norm-powr-real-powr intro!: frac-le
assms powr-mono)
  also have ... = C * (∑ n=1..m. n powr (Re z − 1))
    by (subst sum-distrib-left) (simp-all add: powr-diff)
  also have ... ≤ C * (m powr Re z * (1 / Re z + 1 / m))
    using zeta-partial-sum-le'[of Re z m] assms by (intro mult-left-mono) auto
  finally show ?thesis by (simp add: mult-ac add-ac)
qed

lemma hurwitz-zeta-real-bound-aux:
  fixes a x :: real
  assumes ax: a > 0 x > 1
  shows (∑ i. (a + real (Suc i)) powr (−x)) ≤ a powr (1 − x) / (x − 1)
proof (rule decreasing-sum-le-integral, goal-cases)
  have ((λt. (a + t) powr −x) has-integral −(a powr (−x + 1)) / (−x + 1))
(interior {0..})
   using powr-has-integral-at-top[of 0 a −x] using ax by (simp add: interior-real-atLeast)
  also have −(a powr (− x + 1)) / (− x + 1) = a powr (1 − x) / (x − 1)
    using ax by (simp add: field-simps)
  finally show ((λt. (a + t) powr −x) has-integral a powr (1 − x) / (x − 1))
{0..}
    by (subst (asm) has-integral-interior) auto
qed (insert ax, auto intro!: powr-mono2')
```

Given a function that is analytic on some vertical line segment, we can find

a rectangle around that line segment on which the function is also analytic.

lemma *analytic-on-axis-extend*:
fixes *y1 y2 x* :: *real*
defines $S \equiv \{z.\ Re\ z = x \wedge Im\ z \in \{y1..y2\}\}$
assumes $y1 \leq y2$
assumes *f analytic-on S*
obtains *x1 x2* :: *real* **where** $x1 < x\ x2 > x$ *f analytic-on cbox* (*Complex x1 y1*) (*Complex x2 y2*)
proof −
define *C* **where** $C = \{box\ a\ b\ |a\ b\ z.\ f\ analytic\text{-}on\ box\ a\ b \wedge z \in box\ a\ b \wedge z \in S\}$
have $S = cbox\ (Complex\ x\ y1)\ (Complex\ x\ y2)$
by (*auto simp*: *S-def in-cbox-complex-iff*)
also have *compact* ... **by** *simp*
finally have *1*: *compact S* **.**

have *2*: $S \subseteq \bigcup C$
proof (*intro subsetI*)
fix *z* **assume** $z \in S$
from ‹*f analytic-on S*› **and** *this* **obtain** *a b* **where** $z \in box\ a\ b$ *f analytic-on box a b*
by (*blast elim*: *analytic-onE-box*)
with ‹$z \in S$› **show** $z \in \bigcup C$ **unfolding** *C-def* **by** *blast*
qed

have *3*: *open X* **if** $X \in C$ **for** *X* **using** *that* **by** (*auto simp*: *C-def*)
from *compactE*[*OF 1 2 3*] **obtain** *T* **where** *T*: $T \subseteq C$ *finite* $T\ S \subseteq \bigcup T$
by *blast*

define *x1* **where** $x1 = Max\ (insert\ (x - 1)\ ((\lambda X.\ x + (Inf\ (Re\ `\ X) - x) / 2)\ `\ T))$
define *x2* **where** $x2 = Min\ (insert\ (x + 1)\ ((\lambda X.\ x + (Sup\ (Re\ `\ X) - x) / 2)\ `\ T))$

have ∗: $x + (Inf\ (Re\ `\ X) - x) / 2 < x \wedge x + (Sup\ (Re\ `\ X) - x) / 2 > x$ **if** $X \in T$ **for** *X*
proof −
from *that* **and** *T* **obtain** *a b s* **where** [*simp*]: $X = box\ a\ b$ **and** *s*: $s \in box\ a\ b$ $s \in S$
by (*force simp*: *C-def*)
hence *le*: $Re\ a < Re\ b\ Im\ a < Im\ b$ **by** (*auto simp*: *in-box-complex-iff*)
show *?thesis* **using** *le s*
unfolding ‹$X = box\ a\ b$› *Re-image-box*[*OF le*] *Im-image-box*[*OF le*]
by (*auto simp*: *S-def in-box-complex-iff*)
qed
from ∗ *T* **have** $x1 < x$ **unfolding** *x1-def* **by** (*subst Max-less-iff*) *auto*
from ∗ *T* **have** $x2 > x$ **unfolding** *x2-def* **by** (*subst Min-gr-iff*) *auto*

have *f analytic-on* $(\bigcup T)$

```
    using T by (subst analytic-on-Union) (auto simp: C-def)
  moreover have z ∈ ⋃ T if z ∈ cbox (Complex x1 y1) (Complex x2 y2) for z
  proof −
    from that have Complex x (Im z) ∈ S
      by (auto simp: in-cbox-complex-iff S-def)
    with T obtain X where X: X ∈ T Complex x (Im z) ∈ X
      by auto
    with T obtain a b where [simp]: X = box a b by (auto simp: C-def)
    from X have le: Re a < Re b Im a < Im b by (auto simp: in-box-complex-iff)

    from that have Re z ≤ x2 by (simp add: in-cbox-complex-iff)
    also have ... ≤ x + (Sup (Re ‘ X) − x) / 2
      unfolding x2-def by (rule Min.coboundedI)(use T X in auto)
    also have ... = (x + Re b) / 2
        using le unfolding ‹X = box a b› Re-image-box[OF le] by (simp add:
field-simps)
    also have ... < (Re b + Re b) / 2
      using X by (intro divide-strict-right-mono add-strict-right-mono)
            (auto simp: in-box-complex-iff)
    also have ... = Re b by simp
    finally have [simp]: Re z < Re b .

    have Re a = (Re a + Re a) / 2 by simp
    also have ... < (x + Re a) / 2
      using X by (intro divide-strict-right-mono add-strict-right-mono)
            (auto simp: in-box-complex-iff)
    also have ... = x + (Inf (Re ‘ X) − x) / 2
        using le unfolding ‹X = box a b› Re-image-box[OF le] by (simp add:
field-simps)
    also have ... ≤ x1 unfolding x1-def by (rule Max.coboundedI)(use T X in
auto)
    also have ... ≤ Re z using that by (simp add: in-cbox-complex-iff)
    finally have [simp]: Re z > Re a .

    from X have z ∈ X by (simp add: in-box-complex-iff)
    with T X show ?thesis by blast
  qed
  hence cbox (Complex x1 y1) (Complex x2 y2) ⊆ ⋃ T by blast
  ultimately have f analytic-on cbox (Complex x1 y1) (Complex x2 y2)
    by (rule analytic-on-subset)

  with ‹x1 < x› and ‹x2 > x› and that[of x1 x2] show ?thesis by blast
qed
```

We will now prove the theorem. The precise setting is this: Consider a Dirichlet series $F(s) = \sum a_n n^{-s}$ with bounded coefficients. Clearly, this converges to an analytic function $f(s)$ on $\{s \mid \Re(s) > 1\}$.

If $f(s)$ is analytic on the larger set $\{s \mid \Re(s) \geq 1\}$, F converges to $f(s)$ for all $\Re(s) \geq 1$.

The proof follows Newman's argument very closely, but some of the precise bounds we use are a bit different from his. Also, like Harrison, we choose a combination of a semicircle and a rectangle as our contour, whereas Newman uses a circle with a vertical cut-off. The result of the Residue theorem is the same in both cases, but the bounding of the contributions of the different parts is somewhat different.

The reason why we picked Harrison's contour over Newman's is because we could not understand how his bounding of the different contributions fits to his contour, and it seems likely that this is also the reason why Harrison altered the contour in the first place.

lemma *Newman-Ingham-1*:
  fixes F :: *complex fds* **and** f :: *complex* ⇒ *complex*
  assumes *coeff-bound*: *fds-nth* $F \in O(\lambda\text{-}.\ 1)$
  assumes *f-analytic*: f *analytic-on* $\{s.\ Re\ s \geq 1\}$
  assumes *F-conv-f*: $\bigwedge s.\ Re\ s > 1 \Longrightarrow$ *eval-fds* $F\ s = f\ s$
  assumes *w*: $Re\ w \geq 1$
  shows *fds-converges* $F\ w$ **and** *eval-fds* $F\ w = f\ w$
proof −
  — We get a bound on our coefficients and call it C.
  obtain C **where** C: $C \geq 1\ \bigwedge n.$ *norm* (*fds-nth* $F\ n$) $\leq C$
    using *natfun-bigo-1E*[*OF coeff-bound*, **where** *lb* = *1*] **by** *blast*
  write *contour-integral* (‹$\oint$ [-]›)

  — We show convergence directly by showing that the difference between the partial sums and the limit vanishes.
  have ($\lambda N.$ *eval-fds* (*fds-truncate* $N\ F$) w) ⟶ $f\ w$
    unfolding *tendsto-iff dist-norm norm-minus-commute*[*of eval-fds F s* **for** $F\ s$]
  proof *safe*
    fix ε :: *real* **assume** ε: $\varepsilon > 0$
    — We choose an integration radius that is big enough for the error to be sufficiently small.
    define R **where** $R = max\ 1\ (3 * C\ /\ \varepsilon)$
    have R: $R \geq 3 * C\ /\ \varepsilon\ R \geq 1$ **by** (*auto simp*: *R-def*)

    — Next, we extend the analyticity of $f\ (w + z)$ to the left of the complex plane within a thin rectangle that is at least as high as the circle.
    obtain l **where** l: $l > 0$
      ($\lambda z.\ f\ (w + z)$) *analytic-on* $\{s.\ Re\ s > 0 \vee Im\ s \in \{-R-1<..<R+1\} \wedge Re\ s > -l\}$
    proof −
      have *f-analytic'*: ($\lambda z.\ f\ (w + z)$) *analytic-on* $\{s.\ Re\ s \geq 0\}$
        by (*rule analytic-on-compose-gen*[*OF - f-analytic*, *unfolded o-def*])
          (*insert w*, *auto intro*: *analytic-intros*)
      hence ($\lambda z.\ f\ (w + z)$) *analytic-on* $\{s.\ Re\ s = 0 \wedge Im\ s \in \{-R-1..R+1\}\}$
        by (*rule analytic-on-subset*) *auto*
      from *analytic-on-axis-extend*[*OF - this*] **obtain** *x1 x2* **where** *x12*:
        $x1 < 0\ x2 > 0$ ($\lambda z.\ f\ (w + z)$) *analytic-on cbox* (*Complex x1* $(-R-1)$) (*Complex x2* $(R+1)$)

using ‹$R \geq 1$› **by** *auto*
from *this*(*3*) **have** ($\lambda z.\ f\ (w + z)$) *analytic-on* {*s*. *Re s* ∈ {*x1*..*0*} ∧ *Im s* ∈ {−*R*−*1*..*R*+*1*}}
by (*rule analytic-on-subset*) (*insert x12*, *auto simp*: *in-cbox-complex-iff*)
with *f-analytic'* **have** ($\lambda z.\ f\ (w + z)$) *analytic-on* ({*s*. *Re s* ≥ *0*} ∪ {*s*. *Re s* ∈ {*x1*..*0*} ∧ *Im s* ∈ {−*R*−*1*..*R*+*1*}})
by (*subst analytic-on-Un*) *auto*
hence ($\lambda z.\ f\ (w + z)$) *analytic-on* {*s*. *Re s* > *0* ∨ *Im s* ∈ {−*R*−*1*<..<*R*+*1*} ∧ *Re s* > *x1*}
by (*rule analytic-on-subset*) *auto*
with ‹*x1* < *0*› **and** *that*[*of* −*x1*] **show** *?thesis* **by** *auto*
qed

— The function $f\ (w + z)$ is now analytic on the open box $(-l; R+1)+i(-R+1; R+1)$. We call this region X.
define *X* **where** *X* = *box* (*Complex* (−*l*) (−*R*−*1*)) (*Complex* (*R*+*1*) (*R*+*1*))
have [*simp*, *intro*]: *open X convex X* **by** (*simp-all add*: *X-def open-box*)
from *R l* **have** [*simp*]: *0* ∈ *X* **by** (*auto simp*: *X-def in-box-complex-iff*)
have *analytic*: ($\lambda z.\ f\ (w + z)$) *analytic-on X*
by (*rule analytic-on-subset*[*OF l*(*2*)]) (*auto simp*: *X-def in-box-complex-iff*)
note *f-analytic'* [*analytic-intros*] = *analytic-on-compose-gen*[*OF - analytic*, *unfolded o-def*]
note *f-holo* [*holomorphic-intros*] =
holomorphic-on-compose-gen[*OF - analytic-imp-holomorphic*[*OF analytic*], *unfolded o-def*]
note *f-cont* [*continuous-intros*] = *continuous-on-compose2*[*OF*
holomorphic-on-imp-continuous-on[*OF analytic-imp-holomorphic*[*OF analytic*]]]

— We now pick a smaller closed box X' inside the big open box X. This is because we need a compact set for the next step. our integration path still lies entirely within X', and since X' is compact, $f\ (w + z)$ is bounded on it, so we obtain such a bound and call it M.
define δ **where** δ = *min* (*1*/*2*) (*l*/*2*)
from *l* **have** δ: δ > *0* δ ≤ *1*/*2* δ < *l* **by** (*auto simp*: δ-*def*)
define *X'* **where** *X'* = *cbox* (*Complex* (−δ) (−*R*)) (*Complex R R*)
have *X'* ⊆ *X* **unfolding** *X'-def X-def* **using** *l*(*1*) *R* δ
by (*intro subset-box-imp*) (*auto simp*: *Basis-complex-def*)
have [*intro*]: *compact X'* **by** (*simp add*: *X'-def*)
moreover have *continuous-on X'* ($\lambda z.\ f\ (w + z)$)
using *w* ‹*X'* ⊆ *X*› **by** (*auto intro*!: *continuous-intros*)
ultimately obtain *M* **where** *M*: *M* ≥ *0* ⋀*z*. *z* ∈ *X'* ⟹ *norm* (*f* (*w* + *z*)) ≤ *M*
using *continuous-on-compact-bound* **by** *blast*

— Our objective is now to show that the difference between the N-th partial sum and the limit is below a certain bound (depending on N) which tends to *0* for $N \to \infty$. We use the following bound:

define *bound* **where**
bound = (λ*N*::*nat*. (*2*∗*C*/*R* + *C*/*N* + *3*∗*M* / (*pi*∗*R*∗*ln N*) + *3*∗*R*∗*M* / (δ∗*pi* ∗ *N powr* δ)))
have *2* ∗ *C* / *R* < ε **using** *M*(*1*) *R C*(*1*) δ(*1*) ε
by (*auto simp*: *field-simps*)
— Evidently this is below ε for sufficiently large *N*.
hence *eventually* (λ*N*::*nat*. *bound N* < ε) *at-top*
using *M*(*1*) *R C*(*1*) δ(*1*) ε **unfolding** *bound-def* **by** *real-asymp*

— It now only remains to show that the difference is indeed less than the claimed bound.
thus *eventually* (λ*N*. *norm* (*f w* − *eval-fds* (*fds-truncate N F*) *w*) < ε) *at-top*
using *eventually-gt-at-top*[*of 1*]
proof *eventually-elim*
case (*elim N*)
note *N* = *this*

— Like Harrison (and unlike Newman), our integration path Γ consists of a semicircle *A* of radius *R* in the right-halfplane and a box of width δ and height *2R* on the left halfplane. The latter consists of three straight lines, which we call *B1* to *B3*.
define *A* **where** *A* = *part-circlepath 0 R* (−*pi*/*2*) (*pi*/*2*)
define *B2* **where** *B2* = *linepath* (*Complex* (−δ) *R*) (*Complex* (−δ) (−*R*))
define *B1* **where** *B1* = *linepath* (*R* ∗ i) (*R* ∗ i − δ)
define *B3* **where** *B3* = *linepath* (−*R* ∗ i − δ) (−*R* ∗ i)
define Γ **where** Γ = *A* +++ *B1* +++ *B2* +++ *B3*

— We first need to show some basic facts about the geometry of our integration path.
have [*simp*, *intro*]:
path A path B1 path B3 path B2
valid-path A valid-path B1 valid-path B3 valid-path B2
arc A arc B1 arc B3 arc B2
pathstart A = −i ∗ *R pathfinish A* = i ∗ *R*
pathstart B1 = i ∗ *R pathfinish B1* = *R* ∗ i − δ
pathstart B3 = −*R* ∗ i − δ *pathfinish B3* = −i ∗ *R*
pathstart B2 = *R* ∗ i − δ *pathfinish B2* = −*R* ∗ i − δ **using** *R* δ
by (*simp-all add*: *A-def B1-def B3-def exp-eq-polar B2-def Complex-eq arc-part-circlepath*)
hence [*simp*, *intro*]: *valid-path* Γ
by (*simp add*: Γ-*def A-def B1-def B3-def B2-def exp-eq-polar Complex-eq*)
hence [*simp*, *intro*]: *path* Γ **using** *valid-path-imp-path* **by** *blast*
have [*simp*]: *pathfinish* Γ = *pathstart* Γ **by** (*simp add*: Γ-*def exp-eq-polar*)

have *image-B2*: *path-image B2* = {*s*. *Re s* = −δ ∧ *Im s* ∈ {−*R*..*R*}}
using *R* **by** (*auto simp*: *closed-segment-same-Re closed-segment-eq-real-ivl B2-def*)
have *image-B1*: *path-image B1* = {*s*. *Re s* ∈ {−δ..*0*} ∧ *Im s* = *R*}
and *image-B3*: *path-image B3* = {*s*. *Re s* ∈ {−δ..*0*} ∧ *Im s* = −*R*}

using δ **by** (*auto simp*: *B1-def B3-def closed-segment-same-Im closed-segment-eq-real-ivl*)
have *image-A*: *path-image A* $= \{s.\ Re\ s \geq 0 \wedge norm\ s = R\}$
unfolding *A-def* **using** *R* **by** (*subst path-image-semicircle-Re-ge*) *auto*
also have $z \in \ldots \longrightarrow z \in X' - \{0\}$ **for** *z*
using *complex-Re-le-cmod*[*of z*] *abs-Im-le-cmod*[*of z*] δ *R*
by (*auto simp*: *X'-def in-cbox-complex-iff*)
hence $\{s.\ Re\ s \geq 0 \wedge norm\ s = R\} \subseteq X' - \{0\}$ **by** *auto*
finally have *path-image B2* $\subseteq X' - \{0\}$ *path-image A* $\subseteq X' - \{0\}$
path-image B1 $\subseteq X' - \{0\}$ *path-image B3* $\subseteq X' - \{0\}$ **using** ‹$\delta > 0$›
by (*auto simp*: *X'-def in-cbox-complex-iff image-B2 image-B1 image-B3*)
note *path-images* = *this* ‹$X' \subseteq X$›

— Γ is a simple path, which, combined with its simple geometric shape, makes reasoning about its winding numbers trivial.
from *R* **have** *simple-path A* **unfolding** *A-def*
by (*subst simple-path-part-circlepath*) *auto*
have *simple-path* Γ **unfolding** Γ*-def*
proof (*intro simple-path-join-loop subsetI arc-join*, *goal-cases*)
fix *z* **assume** *z*: $z \in$ *path-image A* $\cap$ *path-image* (*B1 +++ B2 +++ B3*)
with *image-A* **have** *Re z* $\geq$ *0 norm z* = *R* **by** *auto*
with *z R* δ **show** $z \in \{$*pathstart A*, *pathstart* (*B1 +++ B2 +++ B3*)$\}$
by (*auto simp*: *path-image-join image-B1 image-B2 image-B3 complex-eq-iff*)
qed (*insert R*, *auto simp*: *image-B1 image-B3 path-image-join image-B2 complex-eq-iff*)

— We define the integrands in the same fashion as Newman:
define *g* **where** *g* = (λ*z*::*complex*. *f* (*w* + *z*) $*$ *N powr z* $*$ (*1* / *z* + *z* / R^2))
define *S* **where** *S* = *eval-fds* (*fds-truncate N F*)
define *g-S* **where** *g-S* = (λ*z*::*complex*. *S* (*w* + *z*) $*$ *N powr z* $*$ (*1* / *z* + *z* / R^2))
define *rem* **where** *rem* = (λ*z*::*complex*. *f z* − *S z*)
define *g-rem* **where** *g-rem* = (λ*z*::*complex*. *rem* (*w* + *z*) $*$ *N powr z* $*$ (*1* / *z* + *z* / R^2))

have *g-holo*: *g holomorphic-on X* − $\{0\}$ **unfolding** *g-def*
by (*auto intro*!: *holomorphic-intros analytic-imp-holomorphic'*[*OF analytic*])

have *rem-altdef*: *rem z* = *eval-fds* (*fds-remainder N F*) *z* **if** *Re z* > *1* **for** *z*
proof −
have *abscissa*: *abs-conv-abscissa F* $\leq$ *1*
using *assms* **by** (*intro bounded-coeffs-imp-abs-conv-abscissa-le-1*)
(*simp-all add*: *natfun-bigo-iff-Bseq*)
from *assms* **and** *that* **have** *f z* = *eval-fds F z* **by** *auto*
also have *F* = *fds-truncate N F* + *fds-remainder N F*
by (*rule fds-truncate-plus-remainder* [*symmetric*])
also from *that* **have** *eval-fds* ... *z* = *S z* + *eval-fds* (*fds-remainder N F*) *z*
unfolding *S-def*
by (*subst eval-fds-add*) (*auto intro*!: *fds-abs-converges-imp-converges*
fds-abs-converges[*OF le-less-trans*[*OF*

abscissa]])
finally show *?thesis* **by** (*simp add*: *rem-def*)
qed

— We now come to the first application of the residue theorem along the path Γ:
have $\oint [\Gamma]\ g = 2 * pi * \mathrm{i} *$ *winding-number* Γ *0* $*$ *residue g 0*
proof (*subst Residue-theorem*)
show *g holomorphic-on X* − {*0*} **by** *fact*
show *path-image* Γ ⊆ *X* − {*0*} **using** *path-images*
by (*auto simp*: Γ-*def path-image-join*)
thus ∀ *z*. *z* ∉ *X* ⟶ *winding-number* Γ *z* = *0*
by (*auto intro*!: *simply-connected-imp-winding-number-zero*[*of X*]
convex-imp-simply-connected)
qed (*insert path-images*, *auto intro*: *convex-connected*)
also have *winding-number* Γ *0* = *1*
proof (*rule simple-closed-path-winding-number-pos*)
from *R* δ **have** ∀ *g*∈{*A*, *B1*, *B2*, *B3*}. *Re* (*winding-number g 0*) > *0*
unfolding *A-def B1-def B2-def B3-def*
by (*auto intro*!: *winding-number-linepath-pos-lt winding-number-part-circlepath-pos-less*)
hence *valid-path* Γ ∧ *0* ∉ *path-image* Γ ∧ *Re* (*winding-number* Γ *0*) > *0*
unfolding Γ-*def* **using** *path-images*(*1*−*4*) **by** (*intro winding-number-join-pos-combined*′) *auto*
thus *Re* (*winding-number* Γ *0*) > *0* **by** *simp*
qed (*insert path-images* ‹*simple-path* Γ›, *auto simp*: Γ-*def path-image-join*)
also have *residue g 0* = *f w*
proof −
have *g* = (λ*z*::*complex*. *f* (*w* + *z*) $*$ *N powr z* $*$ (*1* + *z*^{2} / *R*^{2}) / *z*)
by (*auto simp*: *g-def divide-simps fun-eq-iff power2-eq-square*
simp del: *div-mult-self3 div-mult-self4 div-mult-self2 div-mult-self1*)
moreover from *N* **have** *residue* ... *0* = *f w*
by (*subst residue-simple*′[*of X*])
(*auto intro*!: *holomorphic-intros analytic-imp-holomorphic*[*OF analytic*])
ultimately show *?thesis* **by** (*simp only*:)
qed
finally have *2* $*$ *pi* $*$ i $*$ *f w* = $\oint [\Gamma]\ g$ **by** *simp*
also have ... = $\oint [A]\ g + \oint [B2]\ g + \oint [B1]\ g + \oint [B3]\ g$ **unfolding** Γ-*def*
by (*subst contour-integral-join*, (*insert path-images*,
auto intro!: *contour-integral-join contour-integrable-holomorphic-simple g-holo*)[*4*])+
(*simp-all add*: *add-ac*)
finally have *integral1*: *2* $*$ *pi* $*$ i $*$ *f w* = $\oint [A]\ g + \oint [B2]\ g + \oint [B1]\ g + \oint [B3]\ g$.

— Next, we apply the residue theorem along a circle of radius R to another integrand that is related to the partial sum:
have $\oint [\mathit{circlepath}\ 0\ R]\ g\text{-}S = 2 * pi * \mathrm{i} *$ *residue g-S 0*
proof (*subst Residue-theorem*)
show *g-S holomorphic-on UNIV* − {*0*}

by (*auto simp*: *g-S-def S-def intro*!: *holomorphic-intros*)
qed (*insert R*, *auto simp*: *winding-number-circlepath-centre*)
also have *residue g-S 0* = *S w*
proof −
have *g-S* = (λ*z*::*complex*. *S* (*w* + *z*) ∗ *N powr z* ∗ (*1* + *z*^{2} / *R*^{2}) / *z*)
by (*auto simp*: *g-S-def divide-simps fun-eq-iff power2-eq-square*
simp del: *div-mult-self3 div-mult-self4 div-mult-self2 div-mult-self1*)
moreover from *N* **have** *residue* ... *0* = *S w*
by (*subst residue-simple*′[*of X*])
(*auto intro*!: *holomorphic-intros simp*: *S-def*)
ultimately show *?thesis* **by** (*simp only*:)
qed
finally have *2* ∗ *pi* ∗ i ∗ *S w* = ∮ [*circlepath 0 R*] *g-S* **..**

— We split this integral into integrals along two semicircles in the left and right half-plane, respectively:
also have ... = ∮ [*part-circlepath 0 R* (−*pi*/*2*) (*3*∗*pi*/*2*)] *g-S*
proof (*rule Cauchy-theorem-homotopic-loops*)
show *homotopic-loops* (−{*0*}) (*circlepath 0 R*)
(*part-circlepath 0 R* (− *pi* / *2*) (*3* ∗ *pi* / *2*)) **unfolding** *circlepath-def* **using** *R*
by (*intro homotopic-loops-part-circlepath*[**where** *k* = *1*]) *auto*
qed (*auto simp*: *g-S-def S-def intro*!: *holomorphic-intros*)
also have ... = ∮ [*A* +++ −*A*] *g-S*
proof (*rule Cauchy-theorem-homotopic-paths*)
have ∗: −*A* = *part-circlepath 0 R* (*pi*/*2*) (*3*∗*pi*/*2*) **unfolding** *A-def*
by (*intro part-circlepath-mirror*[**where** *k* = *0*]) *auto*
from *R* **show** *homotopic-paths* (−{*0*}) (*part-circlepath 0 R* (−*pi*/*2*) (*3*∗*pi*/*2*)) (*A* +++ −*A*)
unfolding ∗ **unfolding** *A-def*
by (*intro homotopic-paths-part-circlepath*) (*auto dest*!: *in-path-image-part-circlepath*)
qed (*auto simp*: *g-S-def S-def A-def exp-eq-polar intro*!: *holomorphic-intros*)
also have ... = ∮ [*A*] *g-S* + ∮ [−*A*] *g-S* **using** *R*
by (*intro contour-integral-join contour-integrable-holomorphic-simple*[*of* - −{*0*}])
(*auto simp*: *A-def g-S-def S-def path-image-mirror dest*!: *in-path-image-part-circlepath intro*!: *holomorphic-intros*)
also have ∮ [−*A*] *g-S* = −∮ [*A*] (λ*x*. *g-S* (−*x*))
by (*simp add*: *A-def contour-integral-mirror contour-integral-neg*)
finally have *integral2*: *2* ∗ *pi* ∗ i ∗ *S w* = ∮ [*A*] *g-S* − ∮ [*A*] (λ*x*. *g-S* (−*x*))
by *simp*

— Next, we show a small bounding lemma that we will need for the final estimate:
have *circle-bound*: *norm* (*1* / *z* + *z* / *R*^{2}) ≤ *2* / *R* **if** [*simp*]: *norm z* = *R* **for** *z* :: *complex*
proof −
have *norm* (*1* / *z* + *z* / *R*^{2}) ≤ *1* / *R* + *1* / *R*
by (*intro order.trans*[*OF norm-triangle-ineq*] *add-mono*)

```
      (insert R, simp-all add: norm-divide norm-mult power2-eq-square)
    thus ?thesis by simp
  qed

  — The next bound differs somewhat from Newman's, but it works just as well.
Its purpose is to bound the contribution of the two short horizontal line segments.
  have B12-bound: norm (integral {− δ..0} (λx. g (x + R′ ∗ i))) ≤ 3 ∗ M /
R / ln N
    (is ?I ≤ -) if |R′| = R for R′
  proof −
    have ?I ≤ integral {−δ..0} (λx. 3 ∗ M / R ∗ N powr x)
    proof (rule integral-norm-bound-integral)
      fix x assume x: x ∈ {−δ..0}
      define z where z = x + i ∗ R′
      from R that have [simp]: z ≠ 0 Re z = x Im z = R′
        by (auto simp: z-def complex-eq-iff)
     from x R that have z ∈ X′ by (auto simp: z-def X′-def in-cbox-complex-iff)
      from x R that have norm z ≤ δ + R
        by (intro order.trans[OF cmod-le add-mono]) auto

      hence norm (1 / z + z / R²) ≤ 1 / R + (δ / R + 1) / R
        using R that abs-Im-le-cmod[of z]
        by (intro order.trans[OF norm-triangle-ineq add-mono])
          (auto simp: norm-divide norm-mult power2-eq-square field-simps )
      also have δ / R ≤ 1 using δ R by auto
      finally have norm (1 / z + z / R²) ≤ 3 / R
        using R by (simp add: divide-right-mono)
      hence norm (g z) ≤ M ∗ N powr x ∗ (3 / R)
        unfolding g-def norm-mult using ‹M ≥ 0› ‹z ∈ X′›
     by (intro mult-mono mult-nonneg-nonneg M) (auto simp: norm-powr-real-powr)
      thus norm (g (x + R′ ∗ i)) ≤ 3 ∗ M / R ∗ N powr x by (simp add:
mult-ac z-def)
    qed (insert N R l that δ, auto intro!: integrable-continuous-real continu-
ous-intros
                    simp: g-def X-def complex-eq-iff in-box-complex-iff)
    also have ... = 3 ∗ M / R ∗ integral {−δ..0} (λx. N powr x) by simp
    also have ((λx. N powr x) has-integral (N powr 0 / ln N − N powr (−δ) /
ln N)) {−δ..0}
      using δ N
      by (intro fundamental-theorem-of-calculus)
         (auto simp: has-real-derivative-iff-has-vector-derivative [symmetric]
powr-def
           intro!: derivative-eq-intros)
    hence integral {−δ..0} (λx. N powr x) = 1 / ln (real N) − real N powr −
δ / ln (real N)
      using N by (simp add: has-integral-iff)
    also have ... ≤ 1 / ln (real N) using N by simp
  finally show ?thesis using M R by (simp add: mult-left-mono divide-right-mono)
  qed
```

— We combine the two results from the residue theorem and obtain an integral representation of the difference between the partial sums and the limit:

have *2 * pi * i * (f w − S w) =*
∮ [A] g − ∮ [A] g-S + ∮ [A] (λx. g-S (−x)) + ∮ [B1] g + ∮ [B3] g + ∮ [B2] g
unfolding *ring-distribs integral1 integral2* **by** (*simp add*: *algebra-simps*)
also have *∮ [A] g − ∮ [A] g-S = ∮ [A] (λx. g x − g-S x)* **using** *path-images*
by (*intro contour-integral-diff* [*symmetric*])
(*auto intro*!: *contour-integrable-holomorphic-simple*[*of - X − {0}*] *holomorphic-intros*
simp: *g-S-def g-holo S-def*)
also have *... = ∮ [A] g-rem*
by (*simp add*: *g-rem-def g-def g-S-def algebra-simps rem-def*)
finally have *2 * pi * i * (f w − S w) =*
∮ [A] g-rem + ∮ [A] (λx. g-S (− x)) + ∮ [B1] g + ∮ [B3] g + ∮ [B2] g **.**

— We now bound each of these integrals individually:

also have *norm ... ≤ 2 * C * pi / R + 2 * C * pi * (1 / N + 1 / R) + 3 * M / R / ln N +*
*3 * M / R / ln N + 6 * R * M * N powr (−δ) / δ*
proof (*rule order.trans*[*OF norm-triangle-ineq*] *add-mono*)+
have *∮ [B1] g = −∮ [reversepath B1] g* **by** (*simp add*: *contour-integral-reversepath*)
also have *∮ [reversepath B1] g = integral {−δ..0} (λx. g (x + R * i))*
unfolding *B1-def reversepath-linepath* **using** *δ*
by (*subst contour-integral-linepath-same-Im*) *auto*
also have *norm (−...) = norm ...* **by** *simp*
also have *... ≤ 3 * M / R / ln N* **using** *R* **by** (*intro B12-bound*) *auto*
finally show *norm (∮ [B1] g) ≤ ...* **.**
next
have *∮ [B3] g = integral {−δ..0} (λx. g (x + (−R) * i))* **unfolding** *B3-def* **using** *δ*
by (*subst contour-integral-linepath-same-Im*) *auto*
also have *norm ... ≤ 3 * M / R / ln N* **using** *R* **by** (*intro B12-bound*) *auto*
finally show *norm (∮ [B3] g) ≤ ...* **.**
next
have *norm (∮ [B2] g) ≤ M * N powr (−δ) * (3 / δ) *
norm (Complex (− δ) (−R) − Complex (− δ) R) **unfolding** *B2-def*
proof ((*rule contour-integral-bound-linepath*; (*fold B2-def*)*?*), *goal-cases*)
case (*3 z*)
from *3 δ R* **have** [*simp*]: *z ≠ 0* **and** *Re-z*: *Re z = −δ* **and** *Im-z*: *Im z ∈ {−R..R}*
by (*auto simp*: *closed-segment-same-Re closed-segment-eq-real-ivl*)
from *3* **have** *z ∈ X′* **using** *R δ path-images* **by** (*auto simp*: *B2-def*)
from *3 δ R* **have** *norm z ≤ sqrt (δ^2 + R^2)* **unfolding** *cmod-def* **using** *Re-z Im-z*
by (*intro real-sqrt-le-mono add-mono*) (*auto simp*: *power2-le-iff-abs-le*)

from *power-mono*[*OF this*, *of 2*] **have** *norm-sqr*: *norm z ^ 2* $\leq \delta^2 + R^2$ **by** *simp*

have *norm* (*1 / z + z / R*^{2}) $\leq$ (*1 + (norm z*)^2 */ R*^{2}) */* δ
unfolding *add-divide-distrib* **using** δ *R abs-Re-le-cmod*[*of z*]
by (*intro order.trans*[*OF norm-triangle-ineq*] *add-mono*)
(*auto simp*: *norm-divide norm-mult field-simps power2-eq-square Re-z*)
also have ... $\leq$ (*1 + (1 +* δ^2 */ R*^{2})) */* δ **using** δ *R* ‹*z* $\in$ *X'*› *norm-sqr* **unfolding** *X'-def*
by (*intro divide-right-mono add-left-mono*)
(*auto simp*: *field-simps in-cbox-complex-iff intro*!: *power-mono*)
also have δ^2 */ R*^{2} $\leq$ *1*
using δ *R* **by** (*auto simp*: *field-simps intro*!: *power-mono*)
finally have *norm* (*1 / z + z / R*^{2}) $\leq$ *3 /* δ **using** δ **by**(*simp add*: *divide-right-mono*)
with ‹*z* $\in$ *X'*› **show** *norm* (*g z*) $\leq$ *M * N powr* ($-\delta$) * (*3 /* δ) **unfolding** *g-def norm-mult*
by (*intro mult-mono mult-nonneg-nonneg M*) (*auto simp*: *norm-powr-real-powr Re-z*)
qed (*insert path-images M* δ, *auto intro*!: *contour-integrable-holomorphic-simple*[*OF g-holo*])
thus *norm* ($\oint$ [*B2*] *g*) $\leq$ *6 * R * M * N powr* ($-\delta$) */* δ
using *R* **by** (*simp add*: *field-simps cmod-def real-sqrt-mult*)
next
have *norm* ($\oint$ [*A*] (λx. *g-S* ($-$ *x*))) $\leq$ (*2 * C / (real N * R) + 2 * C /* R^2) *
R * ((*pi/2*) $-$ ($-$*pi/2*)) **unfolding** *A-def*
proof ((*rule contour-integral-bound-part-circlepath-strong*[**where** *k* = {*R* * i, $-R$*i}];
(*fold A-def*)*?*), *goal-cases*)
case (*6 z*)
hence [*simp*]: *z* $\neq$ *0* **and** *norm z* = *R* **using** *R*
by (*auto simp*: *A-def dest*!: *in-path-image-part-circlepath*)
from *6* **have** *Re z* $\neq$ *0*
using ‹*norm z* = *R*› **by** (*auto simp*: *cmod-def abs-if complex-eq-iff split*: *if-splits*)
with *6* **have** *Re z* > *0* **using** *image-A* **by** *auto*
have *S* (*w* $-$ *z*) = ($\sum$ *k* = *1..N*. *fds-nth F k / of-nat k powr* (*w* $-$ *z*))
by (*simp add*: *S-def eval-fds-truncate*)
also have *norm* ... $\leq$ *C * N powr Re z ** (*1 / N + 1 / Re z*)
using ‹*Re z* > *0*› *w N* **by** (*intro newman-ingham-aux2 C*) *auto*
finally have *norm* (*S* (*w* $-$ *z*)) $\leq$ ... .
hence *norm* (*g-S* ($-$*z*)) $\leq$
(*C * N powr* (*Re z*) * (*1 / N + 1 / Re z*)) * *N powr* ($-$*Re z*) * (*2 * Re z / R*^{2})
unfolding *g-S-def norm-mult*
using *newman-ingham-aux1*[*OF* - ‹*norm z* = *R*›] ‹*Re z* > *0*› ‹*C* $\geq$ *1*› *R*
by (*intro mult-mono mult-nonneg-nonneg circle-bound*)
(*auto simp*: *norm-powr-real-powr norm-uminus-minus*)

```
    also have ... = 2 * C * (Re z / N + 1) / R^2 using R N ‹Re z > 0›
      by (simp add: powr-minus algebra-simps)
   also have ... ≤ 2 * C / (N * R) + 2 * C / R^2 unfolding add-divide-distrib
ring-distribs
      using R N abs-Re-le-cmod[of z] ‹norm z = R› ‹Re z > 0› ‹C ≥ 1›
     by (intro add-mono) (auto simp: power2-eq-square field-simps mult-mono)
     finally show ?case .
  qed (insert R N image-A C, auto intro!: contour-integrable-holomorphic-simple[of
- -{0}]
                                    holomorphic-intros simp: g-S-def S-def)
   also have ... = 2 * C * pi * (1 / N + 1 / R) using R N
     by (simp add: power2-eq-square field-simps)
   finally show norm (∮ [A] (λx. g-S (- x))) ≤ ... .
  next
    have norm (∮ [A] g-rem) ≤ (2 * C / R^2) * R * ((pi/2) - (-pi/2))
unfolding A-def
   proof ((rule contour-integral-bound-part-circlepath-strong[where k = {R *
i, -R*i}];
        (fold A-def)?), goal-cases)
     case (6 z)
     hence [simp]: z ≠ 0 and norm z = R using R
      by (auto simp: A-def dest!: in-path-image-part-circlepath)
     from 6 have Re z ≠ 0
      using ‹norm z = R› by (auto simp: cmod-def abs-if complex-eq-iff split:
if-splits)
     with 6 have Re z > 0 using image-A by auto

     have summable: summable (λn. C * (1 / (Suc n + N) powr (Re w + Re
z)))
      using summable-hurwitz-zeta-real[of Re w + Re z Suc N] ‹Re z > 0› w
     unfolding powr-minus by (intro summable-mult) (auto simp: field-simps)
     have rem (w + z) = (∑ n. fds-nth F (Suc n + N) / (Suc n + N) powr
(w + z))
      using ‹Re z > 0› w by (simp add: rem-altdef eval-fds-remainder)
     also have norm ... ≤ (∑ n. C / (Suc n + N) powr Re (w + z)) using
summable
      by (intro norm-suminf-le)
        (auto simp: norm-divide norm-powr-real-powr intro!: divide-right-mono
C)
     also have ... = (∑ n. C * (Suc n + N) powr -Re (w + z))
      unfolding powr-minus by (simp add: field-simps)
     also have ... = C * (∑ n. (Suc n + N) powr -Re (w + z))
      using summable-hurwitz-zeta-real[of Re w + Re z Suc N] ‹Re z > 0› w
      by (subst suminf-mult) (auto simp: add-ac)
     also have (∑ n. (Suc n + N) powr -Re (w + z)) ≤
                N powr (1 - Re (w + z)) / (Re (w + z) - 1)
      using ‹Re z > 0› w N hurwitz-zeta-real-bound-aux[of N Re (w + z)]
      by (auto simp: add-ac)
     also have ... ≤ N powr -Re z / Re z
```

```
      using w N ‹Re z > 0› by (intro frac-le powr-mono) auto
    finally have norm (rem (w + z)) ≤ C / (Re z * N powr Re z)
      using C by (simp add: mult-left-mono mult-right-mono powr-minus field-simps)
    hence norm (g-rem z) ≤ (C / (Re z * N powr Re z)) * N powr (Re z) * (2 * Re z / R²)
      unfolding g-rem-def norm-mult
      using newman-ingham-aux1[OF - ‹norm z = R›] R ‹Re z > 0› C
      by (intro mult-mono mult-nonneg-nonneg circle-bound)
        (auto simp: norm-powr-real-powr norm-uminus-minus)
    also have ... = 2 * C / R² using R N ‹Re z > 0›
      by (simp add: powr-minus field-simps)
    finally show ?case .
  next
    show g-rem contour-integrable-on A using path-images
      by (auto simp: g-rem-def rem-def S-def
                intro!: contour-integrable-holomorphic-simple[of - X-{0}]
holomorphic-intros)
  qed (insert R N C, auto)
  also have (2 * C / R²) * R * ((pi/2) - (-pi/2)) = 2 * C * pi / R
    using R by (simp add: power2-eq-square field-simps)
  finally show norm (∮ [A] g-rem) ≤ ... .
qed
 also have ... = 4*C*pi/R + 2*C*pi/N + 6*M/R / ln N + 6*R*M*N powr - δ / δ
  by (simp add: algebra-simps)
 also have ... = 2*pi * (2*C/R + C/N + 3*M / (pi*R*ln N) + 3*R*M / (δ*pi * N powr δ))
  by (simp add: field-simps powr-minus )
 also have norm (2 * pi * i * (f w - S w)) = 2 * pi * norm (f w - S w)
  by (simp add: norm-mult)
 finally have norm (f w - S w) ≤ bound N by (simp add: bound-def)
 also have bound N < ε by fact
 finally show norm (f w - S w) < ε .
qed
qed
thus fds-converges F w
  by (auto simp: fds-converges-altdef2 intro: convergentI)
thus eval-fds F w = f w
  using ‹(λN. eval-fds (fds-truncate N F) w) ⟶ f w›
  by (intro tendsto-unique[OF - tendsto-eval-fds-truncate]) auto
qed
```

The theorem generalises in a trivial way; we can replace the requirement that the coefficients of $f(s)$ be $O(1)$ by $O(n^{\sigma-1})$ for some $\sigma \in \mathbb{R}$, then $f(s)$ converges for $\Re(s) > \sigma$. If it can be analytically continued to $\Re(s) \geq \sigma$, it is also convergent there.

```
theorem Newman-Ingham:
  fixes F :: complex fds and f :: complex ⇒ complex
```

assumes *coeff-bound*: *fds-nth F* $\in O(\lambda n.\ n\ powr\ of\text{-}real\ (\sigma - 1))$
assumes *f-analytic*: *f analytic-on* $\{s.\ Re\ s \geq \sigma\}$
assumes *F-conv-f*: $\bigwedge s.\ Re\ s > \sigma \Longrightarrow eval\text{-}fds\ F\ s = f\ s$
assumes *w*: $Re\ w \geq \sigma$
shows *fds-converges F w* **and** $eval\text{-}fds\ F\ w = f\ w$
proof −
define F' **where** $F' = fds\text{-}shift\ (-of\text{-}real\ (\sigma - 1))\ F$
define f' **where** $f' = f \circ (\lambda s.\ s + of\text{-}real\ (\sigma - 1))$

have $fds\text{-}nth\ F' = (\lambda n.\ fds\text{-}nth\ F\ n * of\text{-}nat\ n\ powr\ -of\text{-}real(\sigma - 1))$
by (*auto simp*: *fun-eq-iff F'-def*)
also have $\ldots \in O(\lambda n.\ of\text{-}nat\ n\ powr\ of\text{-}real\ (\sigma - 1) * of\text{-}nat\ n\ powr\ -of\text{-}real(\sigma - 1))$
by (*intro landau-o.big.mult-right assms*)
also have $(\lambda n.\ of\text{-}nat\ n\ powr\ of\text{-}real\ (\sigma - 1) * of\text{-}nat\ n\ powr\ -of\text{-}real\ (\sigma - 1)) \in \Theta(\lambda\text{-}.\ 1)$
by (*intro bigthetaI-cong eventually-mono*[*OF eventually-gt-at-top*[*of 0*]]) (*auto simp*: *powr-minus powr-diff*)
finally have *bigo*: $fds\text{-}nth\ F' \in O(\lambda\text{-}.\ 1)$ **.**

from *f-analytic* **have** *analytic*: f' *analytic-on* $\{s.\ Re\ s \geq 1\}$ **unfolding** *f'-def*
by (*intro analytic-on-compose-gen*[*OF - f-analytic*]) (*auto intro*!: *analytic-intros*)

have *F'-f*: $eval\text{-}fds\ F'\ s = f'\ s$ **if** $Re\ s > 1$ **for** *s*
using *assms that* **by** (*auto simp*: *F'-def f'-def algebra-simps*)

have *w'*: $1 \leq Re\ (w - of\text{-}real\ (\sigma - 1))$
using *w* **by** *simp*

have *1*: $fds\text{-}converges\ F'\ (w - of\text{-}real\ (\sigma - 1))$
using *bigo analytic F'-f w'* **by** (*rule Newman-Ingham-1*)
thus *fds-converges F w* **by** (*auto simp*: *F'-def*)

have *2*: $eval\text{-}fds\ F'\ (w - of\text{-}real\ (\sigma - 1)) = f'\ (w - of\text{-}real\ (\sigma - 1))$
using *bigo analytic F'-f w'* **by** (*rule Newman-Ingham-1*)
thus $eval\text{-}fds\ F\ w = f\ w$
using *assms* **by** (*simp add*: *F'-def f'-def*)
qed

end

3 Prime-Counting Functions

theory *Prime-Counting-Functions*
imports *Prime-Number-Theorem-Library*
begin

We will now define the basic prime-counting functions π, ϑ, and ψ. Additionally, we shall define a function M that is related to Mertens' theorems

and Newman's proof of the Prime Number Theorem. Most of the results in this file are not actually required to prove the Prime Number Theorem, but are still nice to have.

3.1 Definitions

definition *prime-sum-upto* :: $(nat \Rightarrow 'a) \Rightarrow real \Rightarrow 'a$ :: *semiring-1* **where**
prime-sum-upto f x = $(\sum p \mid prime\ p \wedge real\ p \leq x.\ f\ p)$

lemma *prime-sum-upto-altdef1*:
prime-sum-upto f x = *sum-upto* $(\lambda p.\ ind\ prime\ p * f\ p)$ *x*
unfolding *sum-upto-def prime-sum-upto-def*
by (*intro sum.mono-neutral-cong-left finite-subset*[*OF - finite-Nats-le-real*[*of x*]])
(*auto dest*: *prime-gt-1-nat simp*: *ind-def*)

lemma *prime-sum-upto-altdef2*:
prime-sum-upto f x = $(\sum p \mid prime\ p \wedge p \leq nat\ \lfloor x \rfloor.\ f\ p)$
unfolding *sum-upto-altdef prime-sum-upto-altdef1*
by (*intro sum.mono-neutral-cong-right*) (*auto simp*: *ind-def dest*: *prime-gt-1-nat*)

lemma *prime-sum-upto-altdef3*:
prime-sum-upto f x = $(\sum p \leftarrow primes\text{-}upto\ (nat\ \lfloor x \rfloor).\ f\ p)$
proof −
have $(\sum p \leftarrow primes\text{-}upto\ (nat\ \lfloor x \rfloor).\ f\ p) = (\sum p \mid prime\ p \wedge p \leq nat\ \lfloor x \rfloor.\ f\ p)$
by (*subst sum-list-distinct-conv-sum-set*) (*auto simp*: *set-primes-upto conj-commute*)
thus *?thesis* **by** (*simp add*: *prime-sum-upto-altdef2*)
qed

lemma *prime-sum-upto-eqI*:
assumes $a \leq b\ \bigwedge k.\ k \in \{nat\ \lfloor a \rfloor <..nat \lfloor b \rfloor\} \Longrightarrow \neg prime\ k$
shows *prime-sum-upto f a* = *prime-sum-upto f b*
proof −
have ∗: $k \leq nat\ \lfloor a \rfloor$ **if** $k \leq nat\ \lfloor b \rfloor\ prime\ k$ **for** *k*
using *that assms*(*2*)[*of k*] **by** (*cases* $k \leq nat\ \lfloor a \rfloor$) *auto*
from *assms*(*1*) **have** $nat\ \lfloor a \rfloor \leq nat\ \lfloor b \rfloor$ **by** *linarith*
hence $(\sum p \mid prime\ p \wedge p \leq nat\ \lfloor a \rfloor.\ f\ p) = (\sum p \mid prime\ p \wedge p \leq nat\ \lfloor b \rfloor.\ f\ p)$
using *assms* **by** (*intro sum.mono-neutral-left*) (*auto dest*: ∗)
thus *?thesis* **by** (*simp add*: *prime-sum-upto-altdef2*)
qed

lemma *prime-sum-upto-eqI'*:
assumes $a' \leq nat\ \lfloor a \rfloor\ a \leq b\ nat\ \lfloor b \rfloor \leq b'\ \bigwedge k.\ k \in \{a'<..b'\} \Longrightarrow \neg prime\ k$
shows *prime-sum-upto f a* = *prime-sum-upto f b*
by (*rule prime-sum-upto-eqI*) (*use assms* **in** *auto*)

lemmas *eval-prime-sum-upto* = *prime-sum-upto-altdef3*[*unfolded primes-upto-sieve*]

lemma *of-nat-prime-sum-upto*: *of-nat* (*prime-sum-upto f x*) = *prime-sum-upto* $(\lambda p.\ of\text{-}nat\ (f\ p))$ *x*

by (*simp add*: *prime-sum-upto-def*)

lemma *prime-sum-upto-mono*:
assumes $\bigwedge n.\ n > 0 \Longrightarrow f\ n \geq (0::real)$ $x \leq y$
shows *prime-sum-upto f x* ≤ *prime-sum-upto f y*
using *assms* **unfolding** *prime-sum-upto-altdef1 sum-upto-altdef*
by (*intro sum-mono2*) (*auto simp*: *le-nat-iff' le-floor-iff ind-def*)

lemma *prime-sum-upto-nonneg*:
assumes $\bigwedge n.\ n > 0 \Longrightarrow f\ n \geq (0 :: real)$
shows *prime-sum-upto f x* ≥ *0*
unfolding *prime-sum-upto-altdef1 sum-upto-altdef*
by (*intro sum-nonneg*) (*auto simp*: *ind-def assms*)

lemma *prime-sum-upto-eq-0*:
assumes *x* < *2*
shows *prime-sum-upto f x* = *0*
proof −
from *assms* **have** *nat* ⌊*x*⌋ = *0* ∨ *nat* ⌊*x*⌋ = *1* **by** *linarith*
thus *?thesis* **by** (*auto simp*: *eval-prime-sum-upto*)
qed

lemma *measurable-prime-sum-upto* [*measurable*]:
fixes *f* :: *'a* ⇒ *nat* ⇒ *real*
assumes [*measurable*]: $\bigwedge y.\ (\lambda t.\ f\ t\ y) \in M \to_M borel$
assumes [*measurable*]: $x \in M \to_M borel$
shows (λ*t*. *prime-sum-upto* (*f t*) (*x t*)) ∈ *M* →*M* *borel*
unfolding *prime-sum-upto-altdef1* **by** *measurable*

The following theorem breaks down a sum over all prime powers no greater than fixed bound into a nicer form.

lemma *sum-upto-primepows*:
fixes *f* :: *nat* ⇒ *'a* :: *comm-monoid-add*
assumes $\bigwedge n.\ \neg primepow\ n \Longrightarrow f\ n = 0$ $\bigwedge p\ i.\ prime\ p \Longrightarrow i > 0 \Longrightarrow f\ (p\ \hat{}\ i) = g\ p\ i$
shows *sum-upto f x* = (∑ (*p*, *i*) | *prime p* ∧ *i* > *0* ∧ *real* (*p* ^ *i*) ≤ *x*. *g p i*)
proof −
let *?d* = *aprimedivisor*
have *g*: *g* (*?d n*) (*multiplicity* (*?d n*) *n*) = *f n* **if** *primepow n* **for** *n* **using** *that*
by (*subst assms*(*2*) [*symmetric*])
(*auto simp*: *primepow-decompose aprimedivisor-prime-power primepow-gt-Suc-0*
intro!: *aprimedivisor-nat multiplicity-aprimedivisor-gt-0-nat*)
have *sum-upto f x* = (∑ *n* | *primepow n* ∧ *real n* ≤ *x*. *f n*)
unfolding *sum-upto-def* **using** *assms*
by (*intro sum.mono-neutral-cong-right*) (*auto simp*: *primepow-gt-0-nat*)
also have ... = (∑ (*p*, *i*) | *prime p* ∧ *i* > *0* ∧ *real* (*p* ^ *i*) ≤ *x*. *g p i*) (**is** - = *sum* - *?S*)
by (*rule sum.reindex-bij-witness*[*of* - λ(*p*,*i*). *p* ^ *i* λ*n*. (*?d n*, *multiplicity* (*?d n*) *n*)])

(*auto simp*: *aprimedivisor-prime-power primepow-decompose primepow-gt-Suc-0*
g
simp del: *of-nat-power intro*!: *aprimedivisor-nat multiplicity-aprimedivisor-gt-0-nat*)
finally show *?thesis* **.**
qed

definition *primes-pi* **where** *primes-pi* = *prime-sum-upto* (λ*p*. *1* :: *real*)
definition *primes-theta* **where** *primes-theta* = *prime-sum-upto* (λ*p*. *ln* (*real p*))
definition *primes-psi* **where** *primes-psi* = *sum-upto* (*mangoldt* :: *nat* ⇒ *real*)
definition *primes-M* **where** *primes-M* = *prime-sum-upto* (λ*p*. *ln* (*real p*) / *real p*)

Next, we define some nice optional notation for these functions.

open-bundle *prime-counting-syntax*
begin
notation *primes-pi* (‹π›)
notation *primes-theta* (‹ϑ›)
notation *primes-psi* (‹ψ›)
notation *primes-M* (‹𝔐›)
end

lemmas π-*def* = *primes-pi-def*
lemmas ϑ-*def* = *primes-theta-def*
lemmas ψ-*def* = *primes-psi-def*

lemmas *eval-π* = *primes-pi-def*[*unfolded eval-prime-sum-upto*]
lemmas *eval-ϑ* = *primes-theta-def*[*unfolded eval-prime-sum-upto*]
lemmas *eval-𝔐* = *primes-M-def*[*unfolded eval-prime-sum-upto*]

3.2 Basic properties

The proofs in this section are mostly taken from Apostol [1].

lemma *measurable-π* [*measurable*]: π ∈ *borel* →$_M$ *borel*
and *measurable-ϑ* [*measurable*]: ϑ ∈ *borel* →$_M$ *borel*
and *measurable-ψ* [*measurable*]: ψ ∈ *borel* →$_M$ *borel*
and *measurable-primes-M* [*measurable*]: 𝔐 ∈ *borel* →$_M$ *borel*
unfolding *primes-M-def* π-*def* ϑ-*def* ψ-*def* **by** *measurable*

lemma π-*eq-0* [*simp*]: *x* < *2* ⟹ π *x* = *0*
and ϑ-*eq-0* [*simp*]: *x* < *2* ⟹ ϑ *x* = *0*
and *primes-M-eq-0* [*simp*]: *x* < *2* ⟹ 𝔐 *x* = *0*
unfolding *primes-pi-def primes-theta-def primes-M-def*
by (*rule prime-sum-upto-eq-0*; *simp*)+

lemma π-*nat-cancel* [*simp*]: π (*nat x*) = π *x*
and ϑ-*nat-cancel* [*simp*]: ϑ (*nat x*) = ϑ *x*
and *primes-M-nat-cancel* [*simp*]: 𝔐 (*nat x*) = 𝔐 *x*
and ψ-*nat-cancel* [*simp*]: ψ (*nat x*) = ψ *x*

and π-*floor-cancel* [*simp*]: π (*of-int* $\lfloor y \rfloor$) = π y
and ϑ-*floor-cancel* [*simp*]: ϑ (*of-int* $\lfloor y \rfloor$) = ϑ y
and *primes-M-floor-cancel* [*simp*]: $\mathfrak{M}$ (*of-int* $\lfloor y \rfloor$) = $\mathfrak{M}$ y
and ψ-*floor-cancel* [*simp*]: ψ (*of-int* $\lfloor y \rfloor$) = ψ y
by (*simp-all add*: π-*def* ϑ-*def* ψ-*def primes-M-def prime-sum-upto-altdef2 sum-upto-altdef*)

lemma π-*nonneg* [*intro*]: π $x \geq 0$
and ϑ-*nonneg* [*intro*]: ϑ $x \geq 0$
and *primes-M-nonneg* [*intro*]: $\mathfrak{M}$ $x \geq 0$
unfolding *primes-pi-def primes-theta-def primes-M-def*
by (*rule prime-sum-upto-nonneg*; *simp*)+

lemma π-*mono* [*intro*]: $x \leq y \Longrightarrow \pi$ $x \leq \pi$ y
and ϑ-*mono* [*intro*]: $x \leq y \Longrightarrow \vartheta$ $x \leq \vartheta$ y
and *primes-M-mono* [*intro*]: $x \leq y \Longrightarrow \mathfrak{M}$ $x \leq \mathfrak{M}$ y
unfolding *primes-pi-def primes-theta-def primes-M-def*
by (*rule prime-sum-upto-mono*; *simp*)+

lemma π-*pos-iff*: π $x > 0 \longleftrightarrow x \geq 2$
proof
assume x: $x \geq 2$
show π $x > 0$
by (*rule less-le-trans*[*OF* - π-*mono*[*OF* x]]) (*auto simp*: *eval-*π)
next
assume π $x > 0$
hence $\neg(x < 2)$ **by** *auto*
thus $x \geq 2$ **by** *simp*
qed

lemma π-*pos*: $x \geq 2 \Longrightarrow \pi$ $x > 0$
by (*simp add*: π-*pos-iff*)

lemma ψ-*eq-0* [*simp*]:
assumes $x < 2$
shows ψ $x = 0$
proof −
from *assms* **have** *nat* $\lfloor x \rfloor \leq 1$ **by** *linarith*
hence *mangoldt* n = (0 :: *real*) **if** $n \in \{0<..nat \lfloor x \rfloor\}$ **for** n
using *that* **by** (*auto simp*: *mangoldt-def dest*!: *primepow-gt-Suc-0*)
thus *?thesis* **unfolding** ψ-*def sum-upto-altdef* **by** (*intro sum.neutral*) *auto*
qed

lemma ψ-*nonneg* [*intro*]: ψ $x \geq 0$
unfolding ψ-*def sum-upto-def* **by** (*intro sum-nonneg mangoldt-nonneg*)

lemma ψ-*mono*: $x \leq y \Longrightarrow \psi$ $x \leq \psi$ y
unfolding ψ-*def sum-upto-def* **by** (*intro sum-mono2 mangoldt-nonneg*) *auto*

3.3 The n-th prime number

Next we define the n-th prime number, where counting starts from 0. In traditional mathematics, it seems that counting usually starts from 1, but it is more natural to start from 0 in HOL and the asymptotics of the function are the same.

definition *nth-prime* :: *nat* ⇒ *nat* **where**
  nth-prime n = (*THE p. prime p* ∧ *card* {*q. prime q* ∧ *q* < *p*} = *n*)

lemma *finite-primes-less* [*intro*]: *finite* {*q*::*nat. prime q* ∧ *q* < *p*}
  by (*rule finite-subset*[*of* - {..<*p*}]) *auto*

lemma *nth-prime-unique-aux*:
  fixes *p p′* :: *nat*
  assumes *prime p card* {*q. prime q* ∧ *q* < *p*} = *n*
  assumes *prime p′ card* {*q. prime q* ∧ *q* < *p′*} = *n*
  shows *p* = *p′*
  using *assms*
proof (*induction p p′ rule*: *linorder-wlog*)
  case (*le p p′*)
  have *finite* {*q. prime q* ∧ *q* < *p′*} **by** (*rule finite-primes-less*)
  moreover from *le* **have** {*q. prime q* ∧ *q* < *p*} ⊆ {*q. prime q* ∧ *q* < *p′*}
    by *auto*
  moreover from *le* **have** *card* {*q. prime q* ∧ *q* < *p*} = *card* {*q. prime q* ∧ *q* < *p′*}
    by *simp*
  ultimately have {*q. prime q* ∧ *q* < *p*} = {*q. prime q* ∧ *q* < *p′*}
    by (*rule card-subset-eq*)
  with ‹*prime p*› **have** ¬(*p* < *p′*) **by** *blast*
  with ‹*p* ≤ *p′*› **show** *p* = *p′* **by** *auto*
qed *auto*

lemma *π-smallest-prime-beyond*:
  π (*real* (*smallest-prime-beyond m*)) = *π* (*real* (*m* − *1*)) + *1*
proof (*cases m*)
  case *0*
  have *smallest-prime-beyond 0* = *2*
    by (*rule smallest-prime-beyond-eq*) (*auto dest*: *prime-gt-1-nat*)
  with *0* **show** *?thesis* **by** (*simp add*: *eval-π*)
next
  case (*Suc n*)
  define *n′* **where** *n′* = *smallest-prime-beyond* (*Suc n*)
  have *n* < *n′*
    using *smallest-prime-beyond-le*[*of Suc n*] **unfolding** *n′-def* **by** *linarith*
  have *prime n′* **by** (*simp add*: *n′-def*)
  have *n′* ≤ *p* **if** *prime p p* > *n* **for** *p*
    using *that smallest-prime-beyond-smallest*[*of p Suc n*] **by** (*auto simp*: *n′-def*)
  note *n′* = ‹*n* < *n′*› ‹*prime n′*› *this*

```
  have π (real n') = real (card {p. prime p ∧ p ≤ n'})
    by (simp add: π-def prime-sum-upto-def)
  also have Suc n ≤ n' unfolding n'-def by (rule smallest-prime-beyond-le)
  hence {p. prime p ∧ p ≤ n'} = {p. prime p ∧ p ≤ n} ∪ {p. prime p ∧ p ∈
{n<..n'}}
    by auto
  also have real (card ...) = π (real n) + real (card {p. prime p ∧ p ∈ {n<..n'}})
    by (subst card-Un-disjoint) (auto simp: π-def prime-sum-upto-def)
  also have {p. prime p ∧ p ∈ {n<..n'}} = {n'}
    using n' by (auto intro: antisym)
  finally show ?thesis using Suc by (simp add: n'-def)
qed

lemma π-inverse-exists: ∃ n. π (real n) = real m
proof (induction m)
  case 0
  show ?case by (intro exI[of - 0]) auto
next
  case (Suc m)
  from Suc.IH obtain n where n: π (real n) = real m
    by auto
  hence π (real (smallest-prime-beyond (Suc n))) = real (Suc m)
    by (subst π-smallest-prime-beyond) auto
  thus ?case by blast
qed

lemma nth-prime-exists: ∃ p::nat. prime p ∧ card {q. prime q ∧ q < p} = n
proof −
  from π-inverse-exists[of n] obtain m where π (real m) = real n by blast
  hence card: card {q. prime q ∧ q ≤ m} = n
    by (auto simp: π-def prime-sum-upto-def)

  define p where p = smallest-prime-beyond (Suc m)
  have m < p using smallest-prime-beyond-le[of Suc m] unfolding p-def by
linarith
  have prime p by (simp add: p-def)
  have p ≤ q if prime q q > m for q
    using smallest-prime-beyond-smallest[of q Suc m] that by (simp add: p-def)
  note p = ‹m < p› ‹prime p› this

  have {q. prime q ∧ q < p} = {q. prime q ∧ q ≤ m}
  proof safe
    fix q assume prime q q < p
    hence ¬(q > m) using p(1,2) p(3)[of q] by auto
    thus q ≤ m by simp
  qed (insert p, auto)
  also have card ... = n by fact
  finally show ?thesis using ‹prime p› by blast
qed
```

lemma *nth-prime-exists1*: $\exists!p{::}nat.\ prime\ p \wedge card\ \{q.\ prime\ q \wedge q < p\} = n$
by (*intro ex-ex1I nth-prime-exists*) (*blast intro*: *nth-prime-unique-aux*)

lemma *prime-nth-prime* [*intro*]: *prime* (*nth-prime n*)
and *card-less-nth-prime* [*simp*]: *card* $\{q.\ prime\ q \wedge q < nth\text{-}prime\ n\} = n$
using *theI'*[*OF nth-prime-exists1*[*of n*]] **by** (*simp-all add*: *nth-prime-def*)

lemma *card-le-nth-prime* [*simp*]: *card* $\{q.\ prime\ q \wedge q \leq nth\text{-}prime\ n\} = Suc\ n$
proof −
have $\{q.\ prime\ q \wedge q \leq nth\text{-}prime\ n\} = insert\ (nth\text{-}prime\ n)\ \{q.\ prime\ q \wedge q < nth\text{-}prime\ n\}$
by *auto*
also have *card* ... = *Suc n* **by** *simp*
finally show *?thesis* **.**
qed

lemma *π-nth-prime* [*simp*]: $\pi\ (real\ (nth\text{-}prime\ n)) = real\ n + 1$
by (*simp add*: *π-def prime-sum-upto-def*)

lemma *nth-prime-eqI*:
assumes *prime p card* $\{q.\ prime\ q \wedge q < p\} = n$
shows *nth-prime n* = *p*
unfolding *nth-prime-def*
by (*rule the1-equality*[*OF nth-prime-exists1*]) (*use assms* **in** *auto*)

lemma *nth-prime-eqI'*:
assumes *prime p card* $\{q.\ prime\ q \wedge q \leq p\} = Suc\ n$
shows *nth-prime n* = *p*
proof (*rule nth-prime-eqI*)
have $\{q.\ prime\ q \wedge q \leq p\} = insert\ p\ \{q.\ prime\ q \wedge q < p\}$
using *assms* **by** *auto*
also have *card* ... = *Suc* (*card* $\{q.\ prime\ q \wedge q < p\}$)
by *simp*
finally show *card* $\{q.\ prime\ q \wedge q < p\} = n$ **using** *assms* **by** *simp*
qed (*use assms* **in** *auto*)

lemma *nth-prime-eqI''*:
assumes *prime p* $\pi\ (real\ p) = real\ n + 1$
shows *nth-prime n* = *p*
proof (*rule nth-prime-eqI'*)
have *real* (*card* $\{q.\ prime\ q \wedge q \leq p\}$) = $\pi\ (real\ p)$
by (*simp add*: *π-def prime-sum-upto-def*)
also have ... = *real* (*Suc n*) **by** (*simp add*: *assms*)
finally show *card* $\{q.\ prime\ q \wedge q \leq p\} = Suc\ n$
by (*simp only*: *of-nat-eq-iff*)
qed *fact+*

lemma *nth-prime-0* [*simp*]: *nth-prime 0* = *2*

by (*intro nth-prime-eqI*) (*auto dest*: *prime-gt-1-nat*)

lemma *nth-prime-Suc*: *nth-prime* (*Suc n*) = *smallest-prime-beyond* (*Suc* (*nth-prime n*))
by (*rule nth-prime-eqI''*) (*simp-all add*: *π-smallest-prime-beyond*)

lemmas *nth-prime-code* [*code*] = *nth-prime-0 nth-prime-Suc*

lemma *strict-mono-nth-prime*: *strict-mono nth-prime*
proof (*rule strict-monoI-Suc*)
fix *n* :: *nat*
have *Suc* (*nth-prime n*) ≤ *smallest-prime-beyond* (*Suc* (*nth-prime n*)) **by** *simp*
also have ... = *nth-prime* (*Suc n*) **by** (*simp add*: *nth-prime-Suc*)
finally show *nth-prime n* < *nth-prime* (*Suc n*) **by** *simp*
qed

lemma *nth-prime-le-iff* [*simp*]: *nth-prime m* ≤ *nth-prime n* ⟷ *m* ≤ *n*
using *strict-mono-less-eq*[*OF strict-mono-nth-prime*] **by** *blast*

lemma *nth-prime-less-iff* [*simp*]: *nth-prime m* < *nth-prime n* ⟷ *m* < *n*
using *strict-mono-less*[*OF strict-mono-nth-prime*] **by** *blast*

lemma *nth-prime-eq-iff* [*simp*]: *nth-prime m* = *nth-prime n* ⟷ *m* = *n*
using *strict-mono-eq*[*OF strict-mono-nth-prime*] **by** *blast*

lemma *nth-prime-ge-2*: *nth-prime n* ≥ *2*
using *nth-prime-le-iff*[*of 0 n*] **by** (*simp del*: *nth-prime-le-iff*)

lemma *nth-prime-lower-bound*: *nth-prime n* ≥ *Suc* (*Suc n*)
proof −
have *n* = *card* {*q*. *prime q* ∧ *q* < *nth-prime n*}
by *simp*
also have ... ≤ *card* {*2*..<*nth-prime n*}
by (*intro card-mono*) (*auto dest*: *prime-gt-1-nat*)
also have ... = *nth-prime n* − *2* **by** *simp*
finally show *?thesis* **using** *nth-prime-ge-2*[*of n*] **by** *linarith*
qed

lemma *nth-prime-at-top*: *filterlim nth-prime at-top at-top*
proof (*rule filterlim-at-top-mono*)
show *filterlim* (*λn*::*nat*. *n* + *2*) *at-top at-top* **by** *real-asymp*
qed (*auto simp*: *nth-prime-lower-bound*)

lemma *π-at-top*: *filterlim π at-top at-top*
unfolding *filterlim-at-top*
proof *safe*
fix *C* :: *real*
define *x0* **where** *x0* = *real* (*nth-prime* (*nat* ⌈*max 0 C*⌉))
show *eventually* (*λx*. *π x* ≥ *C*) *at-top*

```
    using eventually-ge-at-top
  proof eventually-elim
    fix x assume x ≥ x0
    have C ≤ real (nat ⌈max 0 C⌉ + 1) by linarith
    also have real (nat ⌈max 0 C⌉ + 1) = π x0
      unfolding x0-def by simp
    also have ... ≤ π x by (rule π-mono) fact
    finally show π x ≥ C .
  qed
qed
```

An unbounded, strictly increasing sequence a_n partitions $[a_0; \infty)$ into segments of the form $[a_n; a_{n+1})$.

```
lemma strict-mono-sequence-partition:
  assumes strict-mono (f :: nat ⇒ 'a :: {linorder, no-top})
  assumes x ≥ f 0
  assumes filterlim f at-top at-top
  shows   ∃k. x ∈ {f k..<f (Suc k)}
proof −
  define k where k = (LEAST k. f (Suc k) > x)
  {
    obtain n where x ≤ f n
      using assms by (auto simp: filterlim-at-top eventually-at-top-linorder)
    also have f n < f (Suc n)
      using assms by (auto simp: strict-mono-Suc-iff)
    finally have ∃n. f (Suc n) > x by auto
  }
  from LeastI-ex[OF this] have x < f (Suc k)
    by (simp add: k-def)
  moreover have f k ≤ x
  proof (cases k)
    case (Suc k')
    have k ≤ k' if f (Suc k') > x
      using that unfolding k-def by (rule Least-le)
    with Suc show f k ≤ x by (cases f k ≤ x) (auto simp: not-le)
  qed (use assms in auto)
  ultimately show ?thesis by auto
qed

lemma nth-prime-partition:
  assumes x ≥ 2
  shows   ∃k. x ∈ {nth-prime k..<nth-prime (Suc k)}
 using strict-mono-sequence-partition[OF strict-mono-nth-prime, of x] assms nth-prime-at-top
  by simp

lemma nth-prime-partition':
  assumes x ≥ 2
  shows   ∃k. x ∈ {real (nth-prime k)..<real (nth-prime (Suc k))}
  by (rule strict-mono-sequence-partition)
```

(*auto simp*: *strict-mono-Suc-iff assms*
intro!: *filterlim-real-sequentially filterlim-compose*[*OF - nth-prime-at-top*])

lemma *between-nth-primes-imp-nonprime*:
assumes $n > \textit{nth-prime}\ k\ n < \textit{nth-prime}\ (\textit{Suc}\ k)$
shows $\neg \textit{prime}\ n$
using *assms* **by** (*metis Suc-leI not-le nth-prime-Suc smallest-prime-beyond-smallest*)

lemma *nth-prime-partition''*:
assumes $x \geq (2 :: \textit{real})$
shows $x \in \{\textit{real}\ (\textit{nth-prime}\ (\textit{nat}\ \lfloor \pi\ x \rfloor - 1))..<\textit{real}\ (\textit{nth-prime}\ (\textit{nat}\ \lfloor \pi\ x \rfloor))\}$
proof −
obtain *n* **where** *n*: $x \in \{\textit{nth-prime}\ n..<\textit{nth-prime}\ (\textit{Suc}\ n)\}$
using *nth-prime-partition' assms* **by** *auto*
have $\pi\ (\textit{nth-prime}\ n) = \pi\ x$
unfolding *π-def* **using** *between-nth-primes-imp-nonprime n*
by (*intro prime-sum-upto-eqI*) (*auto simp*: *le-nat-iff le-floor-iff*)
hence $\textit{real}\ n = \pi\ x - 1$
by *simp*
hence *n-eq*: $n = \textit{nat}\ \lfloor \pi\ x \rfloor - 1\ \textit{Suc}\ n = \textit{nat}\ \lfloor \pi\ x \rfloor$
by *linarith*+
with *n* **show** *?thesis*
by *simp*
qed

3.4 Relations between different prime-counting functions

The ψ function can be expressed as a sum of ϑ.

lemma *ψ-altdef*:
assumes $x > 0$
shows $\psi\ x = \textit{sum-upto}\ (\lambda m.\ \textit{prime-sum-upto}\ \textit{ln}\ (\textit{root}\ m\ x))\ (\textit{log}\ 2\ x)$ (**is** - = *?rhs*)
proof −
have *finite*: $\textit{finite}\ \{p.\ \textit{prime}\ p \wedge \textit{real}\ p \leq y\}$ **for** *y*
by (*rule finite-subset*[*of* - {..*nat* ⌊*y*⌋}]) (*auto simp*: *le-nat-iff' le-floor-iff*)
define *S* **where** $S = (\textit{SIGMA}\ i{:}\{i.\ 0 < i \wedge \textit{real}\ i \leq \textit{log}\ 2\ x\}.\ \{p.\ \textit{prime}\ p \wedge \textit{real}\ p \leq \textit{root}\ i\ x\})$
have $\psi\ x = (\sum (p,\ i) \mid \textit{prime}\ p \wedge 0 < i \wedge \textit{real}\ (p\ \hat{}\ i) \leq x.\ \textit{ln}\ (\textit{real}\ p))$ **unfolding** *ψ-def*
by (*subst sum-upto-primepows*[**where** $g = \lambda p\ i.\ \textit{ln}\ (\textit{real}\ p)$])
(*auto simp*: *case-prod-unfold mangoldt-non-primepow*)
also have $\ldots = (\sum (i,\ p) \mid \textit{prime}\ p \wedge 0 < i \wedge \textit{real}\ (p\ \hat{}\ i) \leq x.\ \textit{ln}\ (\textit{real}\ p))$
by (*intro sum.reindex-bij-witness*[*of* - λ(*x*,*y*). (*y*,*x*) λ(*x*,*y*). (*y*,*x*)]) *auto*
also have $\{(i,\ p).\ \textit{prime}\ p \wedge 0 < i \wedge \textit{real}\ (p\ \hat{}\ i) \leq x\} = S$
unfolding *S-def*
proof *safe*
fix *i p* :: *nat* **assume** *ip*: $i > 0\ \textit{real}\ i \leq \textit{log}\ 2\ x\ \textit{prime}\ p\ \textit{real}\ p \leq \textit{root}\ i\ x$
hence $\textit{real}\ (p\ \hat{}\ i) \leq \textit{root}\ i\ x\ \hat{}\ i$ **unfolding** *of-nat-power* **by** (*intro power-mono*) *auto*

```
    with ip assms show real (p ^ i) ≤ x by simp
  next
    fix i p assume ip: prime p i > 0 real (p ^ i) ≤ x
    from ip have 2 ^ i ≤ p ^ i by (intro power-mono) (auto dest: prime-gt-1-nat)
    also have ... ≤ x using ip by simp
    finally show real i ≤ log 2 x
      using assms by (simp add: le-log-iff powr-realpow)
    have root i (real p ^ i) ≤ root i x using ip assms
      by (subst real-root-le-iff) auto
    also have root i (real p ^ i) = real p
      using assms ip by (subst real-root-pos2) auto
    finally show real p ≤ root i x .
  qed
  also have (∑(i,p)∈S. ln p) = sum-upto (λm. prime-sum-upto ln (root m x))
(log 2 x)
    unfolding sum-upto-def prime-sum-upto-def S-def using finite by (subst sum.Sigma)
auto
  finally show ?thesis .
qed

lemma ψ-conv-ϑ-sum: x > 0 ⟹ ψ x = sum-upto (λm. ϑ (root m x)) (log 2 x)
  by (simp add: ψ-altdef ϑ-def)

lemma ψ-minus-ϑ:
  assumes x: x ≥ 2
  shows   ψ x - ϑ x = (∑ i | 2 ≤ i ∧ real i ≤ log 2 x. ϑ (root i x))
proof -
  have finite: finite {i. 2 ≤ i ∧ real i ≤ log 2 x}
    by (rule finite-subset[of - {2..nat ⌊log 2 x⌋}]) (auto simp: le-nat-iff' le-floor-iff)
  have ψ x = (∑ i | 0 < i ∧ real i ≤ log 2 x. ϑ (root i x)) using x
    by (simp add: ψ-conv-ϑ-sum sum-upto-def)
  also have {i. 0 < i ∧ real i ≤ log 2 x} = insert 1 {i. 2 ≤ i ∧ real i ≤ log 2 x}
using x
    by (auto simp: le-log-iff)
  also have (∑ i∈... ϑ (root i x)) - ϑ x =
              (∑ i | 2 ≤ i ∧ real i ≤ log 2 x. ϑ (root i x)) using finite
    by (subst sum.insert) auto
  finally show ?thesis .
qed
```

The following theorems use summation by parts to relate different prime-counting functions to one another with an integral as a remainder term.

```
lemma ϑ-conv-π-integral:
  assumes x ≥ 2
  shows   ((λt. π t / t) has-integral (π x * ln x - ϑ x)) {2..x}
proof (cases x = 2)
  case False
  note [intro] = finite-vimage-real-of-nat-greaterThanAtMost
  from False and assms have x: x > 2 by simp
```

```
  have ((λt. sum-upto (ind prime) t * (1 / t)) has-integral
        sum-upto (ind prime) x * ln x − sum-upto (ind prime) 2 * ln 2 −
        (∑ n∈real −‘ {2<..x}. ind prime n * ln (real n))) {2..x} using x
    by (intro partial-summation-strong[where X = {}])
       (auto intro!: continuous-intros derivative-eq-intros
             simp flip: has-real-derivative-iff-has-vector-derivative)
  hence ((λt. π t / t) has-integral (π x * ln x −
         (π 2 * ln 2 + (∑ n∈real −‘ {2<..x}. ind prime n * ln n)))) {2..x}
    by (simp add: π-def prime-sum-upto-altdef1 algebra-simps)
  also have π 2 * ln 2 + (∑ n∈real −‘ {2<..x}. ind prime n * ln n) =
              (∑ n∈insert 2 (real −‘ {2<..x}). ind prime n * ln n)
    by (subst sum.insert) (auto simp: eval-π)
  also have ... = ϑ x unfolding ϑ-def prime-sum-upto-def using x
   by (intro sum.mono-neutral-cong-right) (auto simp: ind-def dest: prime-gt-1-nat)
  finally show ?thesis .
qed (auto simp: has-integral-refl eval-π eval-ϑ)

lemma π-conv-ϑ-integral:
  assumes x ≥ 2
  shows   ((λt. ϑ t / (t * ln t ^ 2)) has-integral (π x − ϑ x / ln x)) {2..x}
proof (cases x = 2)
  case False
  define b where b = (λp. ind prime p * ln (real p))
  note [intro] = finite-vimage-real-of-nat-greaterThanAtMost
  from False and assms have x: x > 2 by simp
  have ((λt. −(sum-upto b t * (−1 / (t * (ln t)^2)))) has-integral
        −(sum-upto b x * (1 / ln x) − sum-upto b 2 * (1 / ln 2) −
           (∑ n∈real −‘ {2<..x}. b n * (1 / ln (real n))))) {2..x} using x
    by (intro has-integral-neg partial-summation-strong[where X = {}])
       (auto intro!: continuous-intros derivative-eq-intros
         simp flip: has-real-derivative-iff-has-vector-derivative simp add: power2-eq-square)
  also have sum-upto b = ϑ
    by (simp add: ϑ-def b-def prime-sum-upto-altdef1 fun-eq-iff)
  also have ϑ x * (1 / ln x) − ϑ 2 * (1 / ln 2) −
                (∑ n∈real −‘ {2<..x}. b n * (1 / ln (real n))) =
              ϑ x * (1 / ln x) − (∑ n∈insert 2 (real −‘ {2<..x}). b n * (1 / ln
(real n)))
    by (subst sum.insert) (auto simp: b-def eval-ϑ)
  also have (∑ n∈insert 2 (real −‘ {2<..x}). b n * (1 / ln (real n))) = π x using
x
    unfolding π-def prime-sum-upto-altdef1 sum-upto-def
  proof (intro sum.mono-neutral-cong-left ballI, goal-cases)
    case (3 p)
    hence p = 1 by auto
    thus ?case by auto
  qed (auto simp: b-def)
  finally show ?thesis by simp
qed (auto simp: has-integral-refl eval-π eval-ϑ)
```

lemma *integrable-weighted-ϑ*:
assumes $2 \leq a$ $a \leq x$
shows $((\lambda t.\ \vartheta\ t\ /\ (t * ln\ t\ \widehat{}\ 2))$ *integrable-on* $\{a..x\})$
proof (*cases* $a < x$)
case *True*
hence $((\lambda t.\ \vartheta\ t * (1\ /\ (t * ln\ t\ \widehat{}\ 2)))$ *integrable-on* $\{a..x\})$ **using** *assms*
unfolding *ϑ-def prime-sum-upto-altdef1*
by (*intro partial-summation-integrable-strong*[**where** $X = \{\}$ **and** $f = \lambda x.\ -1$ / *ln x*])
(*auto simp flip*: *has-real-derivative-iff-has-vector-derivative*
intro!: *derivative-eq-intros continuous-intros simp*: *power2-eq-square field-simps*)
thus *?thesis* **by** *simp*
qed (*insert has-integral-refl*[*of* - *a*] *assms*, *auto simp*: *has-integral-iff*)

lemma *ϑ-conv-𝔐-integral*:
assumes $x \geq 2$
shows $(\mathfrak{M}$ *has-integral* $(\mathfrak{M}\ x * x - \vartheta\ x))\ \{2..x\}$
proof (*cases* $x = 2$)
case *False*
with *assms* **have** *x*: $x > 2$ **by** *simp*
define $b :: nat \Rightarrow real$ **where** $b = (\lambda p.\ ind\ prime\ p * ln\ p\ /\ p)$
note [*intro*] = *finite-vimage-real-of-nat-greaterThanAtMost*
have *prime-le-2*: $p = 2$ **if** $p \leq 2$ *prime p* **for** $p :: nat$
using *that* **by** (*auto simp*: *prime-nat-iff*)

have $((\lambda t.\ sum\text{-}upto\ b\ t * 1)$ *has-integral sum-upto b x* $*$ *x* $-$ *sum-upto b 2* $*$ *2* $-$
$(\sum n{\in}real\ -`\{2<..x\}.\ b\ n * real\ n))\ \{2..x\}$ **using** *x*
by (*intro partial-summation-strong*[*of* {}])
(*auto simp flip*: *has-real-derivative-iff-has-vector-derivative*
intro!: *derivative-eq-intros continuous-intros*)
also have *sum-upto b* $= \mathfrak{M}$
by (*simp add*: *fun-eq-iff primes-M-def b-def prime-sum-upto-altdef1*)
also have $\mathfrak{M}\ x * x - \mathfrak{M}\ 2 * 2 - (\sum n{\in}real\ -`\{2<..x\}.\ b\ n * real\ n) =$
$\mathfrak{M}\ x * x - (\sum n{\in}insert\ 2\ (real\ -`\{2<..x\}).\ b\ n * real\ n)$
by (*subst sum.insert*) (*auto simp*: *eval-𝔐 b-def*)
also have $(\sum n{\in}insert\ 2\ (real\ -`\{2<..x\}).\ b\ n * real\ n) = \vartheta\ x$
unfolding *ϑ-def prime-sum-upto-def* **using** *x*
by (*intro sum.mono-neutral-cong-right*) (*auto simp*: *b-def ind-def not-less prime-le-2*)
finally show *?thesis* **by** *simp*
qed (*auto simp*: *eval-ϑ eval-𝔐*)

lemma *𝔐-conv-ϑ-integral*:
assumes $x \geq 2$
shows $((\lambda t.\ \vartheta\ t\ /\ t^2)$ *has-integral* $(\mathfrak{M}\ x - \vartheta\ x\ /\ x))\ \{2..x\}$
proof (*cases* $x = 2$)
case *False*
with *assms* **have** *x*: $x > 2$ **by** *simp*

```
  define b :: nat ⇒ real where b = (λp. ind prime p * ln p)
  note [intro] = finite-vimage-real-of-nat-greaterThanAtMost
  have prime-le-2: p = 2 if p ≤ 2 prime p for p :: nat
    using that by (auto simp: prime-nat-iff)

  have ((λt. sum-upto b t * (1 / t^2)) has-integral
        sum-upto b x * (-1 / x) - sum-upto b 2 * (-1 / 2) -
        (∑n∈real -` {2<..x}. b n * (-1 / real n))) {2..x} using x
    by (intro partial-summation-strong[of {}])
     (auto simp flip: has-real-derivative-iff-has-vector-derivative simp: power2-eq-square
           intro!: derivative-eq-intros continuous-intros)
  also have sum-upto b = ϑ
    by (simp add: fun-eq-iff ϑ-def b-def prime-sum-upto-altdef1)
  also have ϑ x * (-1 / x) - ϑ 2 * (-1 / 2) - (∑n∈real -` {2<..x}. b n *
(-1 / real n)) =
            -(ϑ x / x - (∑n∈insert 2 (real -` {2<..x}). b n / real n))
    by (subst sum.insert) (auto simp: eval-ϑ b-def sum-negf)
  also have (∑n∈insert 2 (real -` {2<..x}). b n / real n) = 𝔐 x
    unfolding primes-M-def prime-sum-upto-def using x
   by (intro sum.mono-neutral-cong-right) (auto simp: b-def ind-def not-less prime-le-2)
  finally show ?thesis by simp
qed (auto simp: eval-ϑ eval-𝔐)

lemma integrable-primes-M: 𝔐 integrable-on {x..y} if 2 ≤ x for x y :: real
proof -
  have (λx. 𝔐 x * 1) integrable-on {x..y} if 2 ≤ x x < y for x y :: real
    unfolding primes-M-def prime-sum-upto-altdef1 using that
    by (intro partial-summation-integrable-strong[where X = {} and f = λx. x])
      (auto simp flip: has-real-derivative-iff-has-vector-derivative
           intro!: derivative-eq-intros continuous-intros)
 thus ?thesis using that has-integral-refl(2)[of 𝔐 x] by (cases x y rule: linorder-cases)
auto
qed
```

3.5 Bounds

```
lemma ϑ-upper-bound-coarse:
  assumes x ≥ 1
  shows   ϑ x ≤ x * ln x
proof -
   have ϑ x ≤ sum-upto (λ-. ln x) x unfolding ϑ-def prime-sum-upto-altdef1
sum-upto-def
    by (intro sum-mono) (auto simp: ind-def)
  also have ... ≤ real-of-int ⌊x⌋ * ln x using assms
    by (simp add: sum-upto-altdef)
  also have ... ≤ x * ln x using assms by (intro mult-right-mono) auto
  finally show ?thesis .
qed
```

lemma *ϑ-le-ψ*: $\vartheta\ x \leq \psi\ x$
proof (*cases* $x \geq 2$)
case *False*
hence *nat* $\lfloor x \rfloor = 0 \vee$ *nat* $\lfloor x \rfloor = 1$ **by** *linarith*
thus *?thesis* **by** (*auto simp*: *eval-ϑ*)
next
case *True*
hence $\psi\ x - \vartheta\ x = (\sum i \mid 2 \leq i \wedge$ *real* $i \leq$ *log* 2 $x.\ \vartheta$ (*root* $i\ x$))
by (*rule ψ-minus-ϑ*)
also have $\ldots \geq 0$ **by** (*intro sum-nonneg*) *auto*
finally show *?thesis* **by** *simp*
qed

lemma *π-upper-bound-coarse*:
assumes $x \geq 0$
shows $\pi\ x \leq x\ /\ 3 + 2$
proof −
have {*p*. *prime* $p \wedge p \leq$ *nat* $\lfloor x \rfloor$} $\subseteq$ {2, 3} $\cup$ {*p*. $p \neq 1 \wedge$ *odd* $p \wedge \neg 3$ *dvd* $p \wedge p \leq$ *nat* $\lfloor x \rfloor$}
using *primes-dvd-imp-eq*[*of* 2 :: *nat*] *primes-dvd-imp-eq*[*of* 3 :: *nat*] **by** *auto*
also have $\ldots \subseteq$ {2, 3} $\cup$ ((λ*k*. 6∗*k*+1) ‘ {0<..<*nat* $\lfloor (x+5)/6 \rfloor$} $\cup$ (λ*k*. 6∗*k*+5) ‘ {..<*nat* $\lfloor (x+1)/6 \rfloor$})
(**is** - ∪ *?lhs* ⊆ - ∪ *?rhs*)
proof (*intro Un-mono subsetI*)
fix *p* :: *nat* **assume** $p \in$ *?lhs*
hence *p*: $p \neq 1$ *odd* p ¬*3 dvd p* $p \leq$ *nat* $\lfloor x \rfloor$ **by** *auto*
from *p* (1−3) **have** $(\exists k.\ k > 0 \wedge p = 6 * k + 1 \vee p = 6 * k + 5)$ **by** *presburger*
then obtain *k* **where** $k > 0 \wedge p = 6 * k + 1 \vee p = 6 * k + 5$ **by** *blast*
hence $p = 6 * k + 1 \wedge k > 0 \wedge k <$ *nat* $\lfloor (x+5)/6 \rfloor \vee p = 6{*}k{+}5 \wedge k <$ *nat* $\lfloor (x+1)/6 \rfloor$
unfolding *add-divide-distrib* **using** *p*(4) **by** *linarith*
thus $p \in$ *?rhs* **by** *auto*
qed
finally have *subset*: {*p*. *prime* $p \wedge p \leq$ *nat* $\lfloor x \rfloor$} $\subseteq \ldots$ (**is** - ⊆ *?A*) **.**

have $\pi\ x =$ *real* (*card* {*p*. *prime* $p \wedge p \leq$ *nat* $\lfloor x \rfloor$})
by (*simp add*: *π-def prime-sum-upto-altdef2*)
also have *card* {*p*. *prime* $p \wedge p \leq$ *nat* $\lfloor x \rfloor$} ≤ *card ?A*
by (*intro card-mono subset*) *auto*
also have $\ldots \leq 2 +$ (*nat* $\lfloor (x+5)/6 \rfloor - 1 +$ *nat* $\lfloor (x+1)/6 \rfloor$)
by (*intro order.trans*[*OF card-Un-le*] *add-mono order.trans*[*OF card-image-le*]) *auto*
also have $\ldots \leq x\ /\ 3 + 2$
using *assms* **unfolding** *add-divide-distrib* **by** (*cases* $x \geq 1$, *linarith*, *simp*)
finally show *?thesis* **by** *simp*
qed

lemma *le-numeral-iff*: $m \leq$ *numeral* $n \longleftrightarrow m =$ *numeral* $n \vee m \leq$ *pred-numeral*

n
using *numeral-eq-Suc* **by** *presburger*

The following nice proof for the upper bound $\theta(x) \leq \ln 4 \cdot x$ is taken from Otto Forster's lecture notes on Analytic Number Theory [4].

lemma *prod-primes-upto-less*:
defines $F \equiv (\lambda n.\ (\prod \{p{::}nat.\ prime\ p \wedge p \leq n\}))$
shows $n > 0 \Longrightarrow F\ n < 4\ \hat{}\ n$
proof (*induction n rule*: *less-induct*)
case (*less n*)
have $n = 0 \vee n = 1 \vee n = 2 \vee n = 3 \vee even\ n \wedge n \geq 4 \vee odd\ n \wedge n \geq 4$
by *presburger*
then consider $n = 0 \mid n = 1 \mid n = 2 \mid n = 3 \mid even\ n\ n \geq 4 \mid odd\ n\ n \geq 4$
by *metis*
thus *?case*
proof *cases*
assume [*simp*]: $n = 1$
have $*$: $\{p.\ prime\ p \wedge p \leq Suc\ 0\} = \{\}$ **by** (*auto dest*: *prime-gt-1-nat*)
show *?thesis* **by** (*simp add*: *F-def* $*$)
next
assume [*simp*]: $n = 2$
have $*$: $\{p.\ prime\ p \wedge p \leq 2\} = \{2 :: nat\}$
by (*auto simp*: *le-numeral-iff dest*: *prime-gt-1-nat*)
thus *?thesis* **by** (*simp add*: *F-def* $*$)
next
assume [*simp*]: $n = 3$
have $*$: $\{p.\ prime\ p \wedge p \leq 3\} = \{2, 3 :: nat\}$
by (*auto simp*: *le-numeral-iff dest*: *prime-gt-1-nat*)
thus *?thesis* **by** (*simp add*: *F-def* $*$)
next
assume *n*: *even n n* ≥ 4
from *n* **have** $F\ (n - 1) < 4\ \hat{}\ (n - 1)$ **by** (*intro less.IH*) *auto*
also have $prime\ p \wedge p \leq n \longleftrightarrow prime\ p \wedge p \leq n - 1$ **for** *p*
using *n prime-odd-nat*[*of n*] **by** (*cases* $p = n$) *auto*
hence $F\ (n - 1) = F\ n$ **by** (*simp add*: *F-def*)
also have $4\ \hat{}\ (n - 1) \leq (4\ \hat{}\ n :: nat)$ **by** (*intro power-increasing*) *auto*
finally show *?case* **.**
next
assume *n*: *odd n n* ≥ 4
then obtain *k* **where** *k-eq*: $n = Suc\ (2 * k)$ **by** (*auto elim*: *oddE*)
from *n* **have** *k*: $k \geq 2$ **unfolding** *k-eq* **by** *presburger*
have *prime-dvd*: *p dvd* (*n choose k*) **if** *p*: *prime p p* $\in \{k+1<..n\}$ **for** *p*
proof −
from *p k n* **have** *p dvd pochhammer* $(k + 2)$ *k*
unfolding *pochhammer-prod*
by (*subst prime-dvd-prod-iff*)
(*auto intro*!: *bexI*[*of* - $p - k - 2$] *simp*: *k-eq numeral-2-eq-2 Suc-diff-Suc*)
also have *pochhammer* (*real* $(k + 2)$) *k* = *real* ((*n choose k*) $*$ *fact k*)
by (*simp add*: *binomial-gbinomial gbinomial-pochhammer′ k-eq field-simps*)

```
  hence pochhammer (k + 2) k = (n choose k) * fact k
    unfolding pochhammer-of-nat of-nat-eq-iff .
  finally show p dvd (n choose k) using p
    by (auto simp: prime-dvd-fact-iff prime-dvd-mult-nat)
qed

have ∏{p. prime p ∧ p ∈ {k+1<..n}} dvd (n choose k)
proof (rule multiplicity-le-imp-dvd, goal-cases)
  case (2 p)
  thus ?case
  proof (cases p ∈ {k+1<..n})
    case False
    hence multiplicity p (∏{p. prime p ∧ p ∈ {k+1<..n}}) = 0 using 2
   by (subst prime-elem-multiplicity-prod-distrib) (auto simp: prime-multiplicity-other)
    thus ?thesis by auto
  next
    case True
    hence multiplicity p (∏{p. prime p ∧ p ∈ {k+1<..n}}) =
            sum (multiplicity p) {p. prime p ∧ Suc k < p ∧ p ≤ n} using 2
      by (subst prime-elem-multiplicity-prod-distrib) auto
    also have ... = sum (multiplicity p) {p} using True 2
    proof (intro sum.mono-neutral-right ballI)
      fix q :: nat assume q ∈ {p. prime p ∧ Suc k < p ∧ p ≤ n} - {p}
      thus multiplicity p q = 0 using 2
        by (cases p = q) (auto simp: prime-multiplicity-other)
    qed auto
    also have ... = 1 using 2 by simp
    also have 1 ≤ multiplicity p (n choose k)
      using prime-dvd[of p] 2 True by (intro multiplicity-geI) auto
    finally show ?thesis .
  qed
qed auto
hence ∏{p. prime p ∧ p ∈ {k+1<..n}} ≤ (n choose k)
  by (intro dvd-imp-le) (auto simp: k-eq)
also have ... = 1 / 2 * (∑i∈{k, Suc k}. n choose i)
  using central-binomial-odd[of n] by (simp add: k-eq)
also have (∑i∈{k, Suc k}. n choose i) < (∑i∈{0, k, Suc k}. n choose i)
  using k by simp
also have ... ≤ (∑i≤n. n choose i)
  by (intro sum-mono2) (auto simp: k-eq)
also have ... = (1 + 1) ^ n
  using binomial[of 1 1 n] by simp
also have 1 / 2 * ... = real (4 ^ k)
  by (simp add: k-eq power-mult)
finally have less: (∏{p. prime p ∧ p ∈ {k + 1<..n}}) < 4 ^ k
  unfolding of-nat-less-iff by simp

have F n = F (Suc k) * (∏{p. prime p ∧ p ∈ {k+1<..n}}) unfolding F-def
  by (subst prod.union-disjoint [symmetric]) (auto intro!: prod.cong simp: k-eq)
```

```
    also have ... < 4 ^ Suc k * 4 ^ k using n
      by (intro mult-strict-mono less less.IH) (auto simp: k-eq)
    also have ... = 4 ^ (Suc k + k)
      by (simp add: power-add)
    also have Suc k + k = n by (simp add: k-eq)
    finally show ?case .
  qed (insert less.prems, auto)
qed

lemma ϑ-upper-bound:
  assumes x: x ≥ 1
  shows   ϑ x < ln 4 * x
proof −
  have 4 powr (ϑ x / ln 4) = (∏ p | prime p ∧ p ≤ nat ⌊x⌋. 4 powr (log 4 (real
p)))
    by (simp add: ϑ-def powr-sum prime-sum-upto-altdef2 sum-divide-distrib log-def)
  also have ... = (∏ p | prime p ∧ p ≤ nat ⌊x⌋. real p)
    by (intro prod.cong) (auto dest: prime-gt-1-nat)
  also have ... = real (∏ p | prime p ∧ p ≤ nat ⌊x⌋. p)
    by simp
  also have (∏ p | prime p ∧ p ≤ nat ⌊x⌋. p) < 4 ^ nat ⌊x⌋
    using x by (intro prod-primes-upto-less) auto
  also have ... = 4 powr real (nat ⌊x⌋)
    using x by (subst powr-realpow) auto
  also have ... ≤ 4 powr x
    using x by (intro powr-mono) auto
  finally have 4 powr (ϑ x / ln 4) < 4 powr x
    by simp
  thus ϑ x < ln 4 * x
    by (subst (asm) powr-less-cancel-iff) (auto simp: field-simps)
qed

lemma ϑ-bigo: ϑ ∈ O(λx. x)
  by (intro le-imp-bigo-real[of ln 4] eventually-mono[OF eventually-ge-at-top[of 1]]
          less-imp-le[OF ϑ-upper-bound]) auto

lemma ψ-minus-ϑ-bound:
  assumes x: x ≥ 2
  shows   ψ x − ϑ x ≤ 2 * ln x * sqrt x
proof −
  have ψ x − ϑ x = (∑ i | 2 ≤ i ∧ real i ≤ log 2 x. ϑ (root i x)) using x
    by (rule ψ-minus-ϑ)
  also have ... ≤ (∑ i | 2 ≤ i ∧ real i ≤ log 2 x. ln 4 * root i x)
    using x by (intro sum-mono less-imp-le[OF ϑ-upper-bound]) auto
  also have ... ≤ (∑ i | 2 ≤ i ∧ real i ≤ log 2 x. ln 4 * root 2 x) using x
     by (intro sum-mono mult-mono) (auto simp: le-log-iff powr-realpow intro!:
real-root-decreasing)
  also have ... = card {i. 2 ≤ i ∧ real i ≤ log 2 x} * ln 4 * sqrt x
    by (simp add: sqrt-def)
```

```
  also have {i. 2 ≤ i ∧ real i ≤ log 2 x} = {2..nat ⌊log 2 x⌋}
    by (auto simp: le-nat-iff' le-floor-iff)
  also have log 2 x ≥ 1 using x by (simp add: le-log-iff)
  hence real (nat ⌊log 2 x⌋ − 1) ≤ log 2 x using x by linarith
  hence card {2..nat ⌊log 2 x⌋} ≤ log 2 x by simp
  also have ln (2 * 2 :: real) = 2 * ln 2 by (subst ln-mult) auto
  hence log 2 x * ln 4 * sqrt x = 2 * ln x * sqrt x using x
    by (simp add: ln-sqrt log-def power2-eq-square field-simps)
  finally show ?thesis using x by (simp add: mult-right-mono)
qed

lemma ψ-minus-ϑ-bigo: (λx. ψ x − ϑ x) ∈ O(λx. ln x * sqrt x)
proof (intro bigoI[of - 2] eventually-mono[OF eventually-ge-at-top[of 2]])
  fix x :: real assume x ≥ 2
  thus norm (ψ x − ϑ x) ≤ 2 * norm (ln x * sqrt x)
    using ψ-minus-ϑ-bound[of x] ϑ-le-ψ[of x] by simp
qed

lemma ψ-bigo: ψ ∈ O(λx. x)
proof −
  have (λx. ψ x − ϑ x) ∈ O(λx. ln x * sqrt x)
    by (rule ψ-minus-ϑ-bigo)
  also have (λx. ln x * sqrt x) ∈ O(λx. x)
    by real-asymp
  finally have (λx. ψ x − ϑ x + ϑ x) ∈ O(λx. x)
    by (rule sum-in-bigo) (fact ϑ-bigo)
  thus ?thesis by simp
qed
```

We shall now attempt to get some more concrete bounds on the difference between $\pi(x)$ and $\theta(x)/\ln x$ These will be essential in showing the Prime Number Theorem later.

We first need some bounds on the integral

$$\int_2^x \frac{1}{\ln^2 t}\,\mathrm{d}t$$

in order to bound the contribution of the remainder term. This integral actually has an antiderivative in terms of the logarithmic integral $\mathrm{li}(x)$, but since we do not have a formalisation of it in Isabelle, we will instead use the following ad-hoc bound given by Apostol:

```
lemma integral-one-over-log-squared-bound:
  assumes x: x ≥ 4
  shows   integral {2..x} (λt. 1 / ln t ^ 2) ≤ sqrt x / ln 2 ^ 2 + 4 * x / ln x ^ 2
proof −
  from x have x * 1 ≤ x ^ 2 unfolding power2-eq-square by (intro mult-left-mono)
auto
  with x have x': 2 ≤ sqrt x sqrt x ≤ x
    by (auto simp: real-sqrt-le-iff' intro!: real-le-rsqrt)
```

```
  have integral {2..x} (λt. 1 / ln t ^ 2) =
        integral {2..sqrt x} (λt. 1 / ln t ^ 2) + integral {sqrt x..x} (λt. 1 / ln t ^
2)
    (is - = ?I1 + ?I2) using x x'
   by (intro Henstock-Kurzweil-Integration.integral-combine [symmetric] integrable-continuous-real)
      (auto intro!: continuous-intros)
  also have ?I1 ≤ integral {2..sqrt x} (λ-. 1 / ln 2 ^ 2) using x
    by (intro integral-le integrable-continuous-real divide-left-mono
          power-mono continuous-intros) auto
  also have ... ≤ sqrt x / ln 2 ^ 2 using x' by (simp add: field-simps)
  also have ?I2 ≤ integral {sqrt x..x} (λt. 1 / ln (sqrt x) ^ 2) using x'
    by (intro integral-le integrable-continuous-real divide-left-mono
          power-mono continuous-intros) auto
  also have ... ≤ 4 * x / ln x ^ 2 using x' by (simp add: ln-sqrt field-simps)
  finally show ?thesis by simp
qed

lemma integral-one-over-log-squared-bigo:
  (λx::real. integral {2..x} (λt. 1 / ln t ^ 2)) ∈ O(λx. x / ln x ^ 2)
proof −
  define ub where ub = (λx::real. sqrt x / ln 2 ^ 2 + 4 * x / ln x ^ 2)
  have eventually (λx. |integral {2..x} (λt. 1 / (ln t)^2)| ≤ |ub x|) at-top
    using eventually-ge-at-top[of 4]
  proof eventually-elim
    case (elim x)
    hence |integral {2..x} (λt. 1 / ln t ^ 2)| = integral {2..x} (λt. 1 / ln t ^ 2)
      by (intro abs-of-nonneg integral-nonneg integrable-continuous-real continu-
ous-intros) auto
    also have ... ≤ |ub x|
      using integral-one-over-log-squared-bound[of x] elim by (simp add: ub-def)
    finally show ?case .
  qed
  hence (λx. integral {2..x} (λt. 1 / (ln t)^2)) ∈ O(ub)
    by (intro landau-o.bigI[of 1]) auto
  also have ub ∈ O(λx. x / ln x ^ 2) unfolding ub-def by real-asymp
  finally show ?thesis .
qed

lemma π-ϑ-bound:
  assumes x ≥ (4 :: real)
  defines ub ≡ 2 / ln 2 * sqrt x + 8 * ln 2 * x / ln x ^ 2
  shows   π x − ϑ x / ln x ∈ {0..ub}
proof −
  define r where r = (λx. integral {2..x} (λt. ϑ t / (t * ln t ^ 2)))
  have integrable: (λt. c / ln t ^ 2) integrable-on {2..x} for c
    by (intro integrable-continuous-real continuous-intros) auto

  have r x ≤ integral {2..x} (λt. ln 4 / ln t ^ 2) unfolding r-def
     using integrable-weighted-ϑ[of 2 x] integrable[of ln 4] assms less-imp-le[OF
```

ϑ-upper-bound]
  by (*intro integral-le divide-right-mono*) (*auto simp*: *field-simps*)
 also have ... = *ln 4 * integral {2..x}* (λ*t. 1 / ln t ^ 2*)
  using *integrable*[*of 1*] **by** (*subst integral-mult*) *auto*
 also have ... ≤ *ln 4 * (sqrt x / ln 2 ^ 2 + 4 * x / ln x ^ 2)*
  using *assms* **by** (*intro mult-left-mono integral-one-over-log-squared-bound*) *auto*
 also have *ln* (*4* :: *real*) = *2 * ln 2*
  using *ln-realpow*[*of 2 2*] **by** *simp*
 also have ... * (*sqrt x / ln 2 ^ 2 + 4 * x / ln x ^ 2*) = *ub*
  using *assms* **by** (*simp add*: *field-simps power2-eq-square ub-def*)
 finally have *r x* ≤ ... **.**
 moreover have *r x* ≥ *0* **unfolding** *r-def* **using** *assms*
  by (*intro integral-nonneg integrable-weighted-ϑ divide-nonneg-pos*) *auto*
 ultimately have *r x* ∈ {*0..ub*} **by** *auto*
 with *π-conv-ϑ-integral*[*of x*] *assms*(*1*) **show** *?thesis*
  by (*simp add*: *r-def has-integral-iff*)
qed

The following statement already indicates that the asymptotics of π and ϑ are very closely related, since through it, $\pi(x) \sim x/\ln x$ and $\theta(x) \sim x$ imply each other.

lemma *π-ϑ-bigo*: (λ*x. π x − ϑ x / ln x*) ∈ *O*(λ*x. x / ln x ^ 2*)
proof −
 define *ub* **where** *ub* = (λ*x. 2 / ln 2 * sqrt x + 8 * ln 2 * x / ln x ^ 2*)
 have (λ*x. π x − ϑ x / ln x*) ∈ *O*(*ub*)
 proof (*intro le-imp-bigo-real*[*of 1*] *eventually-mono*[*OF eventually-ge-at-top*])
  fix *x* :: *real* **assume** *x* ≥ *4*
  from *π-ϑ-bound*[*OF this*] **show** *π x − ϑ x / ln x* ≥ *0* **and** *π x − ϑ x / ln x* ≤ *1 * ub x*
   by (*simp-all add*: *ub-def*)
 qed *auto*
 also have *ub* ∈ *O*(λ*x. x / ln x ^ 2*)
  unfolding *ub-def* **by** *real-asymp*
 finally show *?thesis* **.**
qed

As a foreshadowing of the Prime Number Theorem, we can already show the following upper bound on $\pi(x)$:

lemma *π-upper-bound*:
 assumes *x* ≥ (*4* :: *real*)
 shows *π x* < *ln 4 * x / ln x + 8 * ln 2 * x / ln x ^ 2 + 2 / ln 2 * sqrt x*
proof −
 define *ub* **where** *ub* = *2 / ln 2 * sqrt x + 8 * ln 2 * x / ln x ^ 2*
 have *π x* ≤ *ϑ x / ln x + ub*
  using *π-ϑ-bound*[*of x*] *assms* **unfolding** *ub-def* **by** *simp*
 also from *assms* **have** *ϑ x / ln x* < *ln 4 * x / ln x*
  by (*intro ϑ-upper-bound divide-strict-right-mono*) *auto*
 finally show *?thesis*
  using *assms* **by** (*simp add*: *algebra-simps ub-def*)

qed

lemma *π-bigo*: $\pi \in O(\lambda x.\ x\ /\ ln\ x)$
proof −
  have $(\lambda x.\ \pi\ x - \vartheta\ x\ /\ ln\ x) \in O(\lambda x.\ x\ /\ ln\ x\ \hat{}\ 2)$
    by (*fact π-ϑ-bigo*)
  also have $(\lambda x{::}real.\ x\ /\ ln\ x\ \hat{}\ 2) \in O(\lambda x.\ x\ /\ ln\ x)$
    by *real-asymp*
  finally have $(\lambda x.\ \pi\ x - \vartheta\ x\ /\ ln\ x) \in O(\lambda x.\ x\ /\ ln\ x)$ **.**
  moreover have *eventually* $(\lambda x{::}real.\ ln\ x > 0)$ *at-top* **by** *real-asymp*
  hence *eventually* $(\lambda x{::}real.\ ln\ x \neq 0)$ *at-top* **by** *eventually-elim auto*
  hence $(\lambda x.\ \vartheta\ x\ /\ ln\ x) \in O(\lambda x.\ x\ /\ ln\ x)$
    using *ϑ-bigo* **by** (*intro landau-o.big.divide-right*)
  ultimately have $(\lambda x.\ \pi\ x - \vartheta\ x\ /\ ln\ x + \vartheta\ x\ /\ ln\ x) \in O(\lambda x.\ x\ /\ ln\ x)$
    by (*rule sum-in-bigo*)
  thus *?thesis* **by** *simp*
qed

3.6 Equivalence of various forms of the Prime Number Theorem

In this section, we show that the following forms of the Prime Number Theorem are all equivalent:

1. $\pi(x) \sim x/\ln x$
2. $\pi(x)\ln\pi(x) \sim x$
3. $p_n \sim n\ln n$
4. $\vartheta(x) \sim x$
5. $\psi(x) \sim x$

We show the following implication chains:

- $(1) \to (2) \to (3) \to (2) \to (1)$
- $(1) \to (4) \to (1)$
- $(4) \to (5) \to (4)$

All of these proofs are taken from Apostol's book.

lemma *PNT1-imp-PNT1′*:
  assumes $\pi \sim[at\text{-}top]\ (\lambda x.\ x\ /\ ln\ x)$
  shows $(\lambda x.\ ln\ (\pi\ x)) \sim[at\text{-}top]\ ln$
proof −

```
  from assms have ((λx. π x / (x / ln x)) ⟶ 1) at-top
    by (rule asymp-equivD-strong[OF - eventually-mono[OF eventually-gt-at-top[of
1]]]) auto
  hence ((λx. ln (π x / (x / ln x))) ⟶ ln 1) at-top
    by (rule tendsto-ln) auto
  also have ?this ⟷ ((λx. ln (π x) − ln x + ln (ln x)) ⟶ 0) at-top
    by (intro filterlim-cong eventually-mono[OF eventually-gt-at-top[of 2]])
      (auto simp: ln-divide-pos field-simps π-pos-iff ln-mult-pos)
  finally have (λx. ln (π x) − ln x + ln (ln x)) ∈ o(λ-. 1)
    by (intro smalloI-tendsto) auto
  also have (λ-::real. 1 :: real) ∈ o(λx. ln x)
    by real-asymp
  finally have (λx. ln (π x) − ln x + ln (ln x) − ln (ln x)) ∈ o(λx. ln x)
    by (rule sum-in-smallo) real-asymp+
  thus ∗: (λx. ln (π x)) ∼[at-top] ln
    by (simp add: asymp-equiv-altdef)
qed

lemma PNT1-imp-PNT2:
  assumes π ∼[at-top] (λx. x / ln x)
  shows   (λx. π x ∗ ln (π x)) ∼[at-top] (λx. x)
proof −
  have (λx. π x ∗ ln (π x)) ∼[at-top] (λx. x / ln x ∗ ln x)
    by (intro asymp-equiv-intros assms PNT1-imp-PNT1')
  also have ... ∼[at-top] (λx. x)
    by (intro asymp-equiv-refl-ev eventually-mono[OF eventually-gt-at-top[of 1]])
      (auto simp: field-simps)
  finally show (λx. π x ∗ ln (π x)) ∼[at-top] (λx. x)
    by simp
qed

lemma PNT2-imp-PNT3:
  assumes (λx. π x ∗ ln (π x)) ∼[at-top] (λx. x)
  shows   nth-prime ∼[at-top] (λn. n ∗ ln n)
proof −
  have (λn. nth-prime n) ∼[at-top] (λn. π (nth-prime n) ∗ ln (π (nth-prime n)))
    using assms
    by (rule asymp-equiv-symI [OF asymp-equiv-compose'])
      (auto intro!: filterlim-compose[OF filterlim-real-sequentially nth-prime-at-top])
  also have ... = (λn. real (Suc n) ∗ ln (real (Suc n)))
    by (simp add: add-ac)
  also have ... ∼[at-top] (λn. real n ∗ ln (real n))
    by real-asymp
  finally show nth-prime ∼[at-top] (λn. n ∗ ln n) .
qed

lemma PNT3-imp-PNT2:
  assumes nth-prime ∼[at-top] (λn. n ∗ ln n)
  shows   (λx. π x ∗ ln (π x)) ∼[at-top] (λx. x)
```

proof (*rule asymp-equiv-symI*, *rule asymp-equiv-sandwich-real*)
show *eventually* (λ*x*. *x* ∈ {*real* (*nth-prime* (*nat* ⌊π *x*⌋ − *1*))..*real* (*nth-prime* (*nat* ⌊π *x*⌋))}) *at-top*
using *eventually-ge-at-top*[*of 2*]
proof *eventually-elim*
case (*elim x*)
with *nth-prime-partition''*[*of x*] **show** *?case* **by** *auto*
qed
next
have (λ*x*. *real* (*nth-prime* (*nat* ⌊π *x*⌋ − *1*))) ∼[*at-top*] (λ*x*. *real* (*nat* ⌊π *x*⌋ − *1*) ∗ *ln* (*real* (*nat* ⌊π *x*⌋ − *1*)))
by (*rule asymp-equiv-compose'*[*OF - π-at-top*], *rule asymp-equiv-compose'*[*OF assms*]) *real-asymp*
also have ... ∼[*at-top*] (λ*x*. π *x* ∗ *ln* (π *x*))
by (*rule asymp-equiv-compose'*[*OF - π-at-top*]) *real-asymp*
finally show (λ*x*. *real* (*nth-prime* (*nat* ⌊π *x*⌋ − *1*))) ∼[*at-top*] (λ*x*. π *x* ∗ *ln* (π *x*)) **.**
next
have (λ*x*. *real* (*nth-prime* (*nat* ⌊π *x*⌋))) ∼[*at-top*] (λ*x*. *real* (*nat* ⌊π *x*⌋) ∗ *ln* (*real* (*nat* ⌊π *x*⌋)))
by (*rule asymp-equiv-compose'*[*OF - π-at-top*], *rule asymp-equiv-compose'*[*OF assms*]) *real-asymp*
also have ... ∼[*at-top*] (λ*x*. π *x* ∗ *ln* (π *x*))
by (*rule asymp-equiv-compose'*[*OF - π-at-top*]) *real-asymp*
finally show (λ*x*. *real* (*nth-prime* (*nat* ⌊π *x*⌋))) ∼[*at-top*] (λ*x*. π *x* ∗ *ln* (π *x*)) **.**
qed

lemma *PNT2-imp-PNT1*:
assumes (λ*x*. π *x* ∗ *ln* (π *x*)) ∼[*at-top*] (λ*x*. *x*)
shows (λ*x*. *ln* (π *x*)) ∼[*at-top*] (λ*x*. *ln x*)
and π ∼[*at-top*] (λ*x*. *x* / *ln x*)
proof −
have *ev*: *eventually* (λ*x*. π *x* > *0*) *at-top*
eventually (λ*x*. *ln* (π *x*) > *0*) *at-top*
eventually (λ*x*. *ln* (*ln* (π *x*)) > *0*) *at-top*
by (*rule eventually-compose-filterlim*[*OF - π-at-top*], *real-asymp*)+

let *?f* = λ*x*. *1* + *ln* (*ln* (π *x*)) / *ln* (π *x*) − *ln x* / *ln* (π *x*)
have ((λ*x*. *ln* (π *x*) ∗ *?f x*) ⟶ *ln 1*) *at-top*
proof (*rule Lim-transform-eventually*)
from *assms* **have** ((λ*x*. π *x* ∗ *ln* (π *x*) / *x*) ⟶ *1*) *at-top*
by (*rule asymp-equivD-strong*[*OF - eventually-mono*[*OF eventually-gt-at-top*[*of 1*]]]) *auto*
then show ((λ*x*. *ln* (π *x* ∗ *ln* (π *x*) / *x*)) ⟶ *ln 1*) *at-top*
by (*rule tendsto-ln*) *auto*
show ∀ *F x in at-top*. *ln* (π *x* ∗ *ln* (π *x*) / *x*) = *ln* (π *x*) ∗ *?f x*
using *eventually-gt-at-top*[*of 0*] *ev*
by *eventually-elim* (*simp add*: *field-simps ln-mult ln-div*)

```
  qed
  moreover have ((λx. 1 / ln (π x)) ⟶ 0) at-top
    by (rule filterlim-compose[OF - π-at-top]) real-asymp
  ultimately have ((λx. ln (π x) * ?f x * (1 / ln (π x))) ⟶ ln 1 * 0) at-top
    by (rule tendsto-mult)
  moreover have eventually (λx. ln (π x) * ?f x * (1 / ln (π x)) = ?f x) at-top
    using ev by eventually-elim auto
  ultimately have (?f ⟶ ln 1 * 0) at-top
    by (rule Lim-transform-eventually)
  hence ((λx. 1 + ln (ln (π x)) / ln (π x) - ?f x) ⟶ 1 + 0 - ln 1 * 0) at-top
    by (intro tendsto-intros filterlim-compose[OF - π-at-top]) (real-asymp | simp)+
  hence ((λx. ln x / ln (π x)) ⟶ 1) at-top
    by simp
  thus *: (λx. ln (π x)) ∼[at-top] (λx. ln x)
    by (rule asymp-equiv-symI[OF asymp-equivI'])

  have eventually (λx. π x = π x * ln (π x) / ln (π x)) at-top
    using ev by eventually-elim auto
  hence π ∼[at-top] (λx. π x * ln (π x) / ln (π x))
    by (rule asymp-equiv-refl-ev)
  also from assms and * have (λx. π x * ln (π x) / ln (π x)) ∼[at-top] (λx. x /
ln x)
    by (rule asymp-equiv-intros)
  finally show π ∼[at-top] (λx. x / ln x) .
qed

lemma PNT4-imp-PNT5:
  assumes ϑ ∼[at-top] (λx. x)
  shows   ψ ∼[at-top] (λx. x)
proof −
  define r where r = (λx. ψ x − ϑ x)
  have r ∈ O(λx. ln x * sqrt x)
    unfolding r-def by (fact ψ-minus-ϑ-bigo)
  also have (λx::real. ln x * sqrt x) ∈ o(λx. x)
    by real-asymp
  finally have r: r ∈ o(λx. x) .

  have (λx. ϑ x + r x) ∼[at-top] (λx. x)
    using assms r by (subst asymp-equiv-add-right) auto
  thus ?thesis by (simp add: r-def)
qed

lemma PNT4-imp-PNT1:
  assumes ϑ ∼[at-top] (λx. x)
  shows   π ∼[at-top] (λx. x / ln x)
proof −
  have (λx. (π x − ϑ x / ln x) + ((ϑ x − x) / ln x)) ∈ o(λx. x / ln x)
  proof (rule sum-in-smallo)
    have (λx. π x − ϑ x / ln x) ∈ O(λx. x / ln x ^ 2)
```

```
    by (rule π-ϑ-bigo)
  also have (λx. x / ln x ^ 2) ∈ o(λx. x / ln x :: real)
    by real-asymp
  finally show (λx. π x − ϑ x / ln x) ∈ o(λx. x / ln x) .
 next
  have eventually (λx::real. ln x > 0) at-top by real-asymp
  hence eventually (λx::real. ln x ≠ 0) at-top by eventually-elim auto
  thus (λx. (ϑ x − x) / ln x) ∈ o(λx. x / ln x)
    by (intro landau-o.small.divide-right asymp-equiv-imp-diff-smallo assms)
 qed
 thus ?thesis by (simp add: diff-divide-distrib asymp-equiv-altdef)
qed

lemma PNT1-imp-PNT4:
 assumes π ∼[at-top] (λx. x / ln x)
 shows   ϑ ∼[at-top] (λx. x)
proof −
 have ϑ ∼[at-top] (λx. π x * ln x)
 proof (rule smallo-imp-asymp-equiv)
  have (λx. ϑ x − π x * ln x) ∈ Θ(λx. − ((π x − ϑ x / ln x) * ln x))
    by (intro bigthetaI-cong eventually-mono[OF eventually-gt-at-top[of 1]])
       (auto simp: field-simps)
  also have (λx. − ((π x − ϑ x / ln x) * ln x)) ∈ O(λx. x / (ln x)^2 * ln x)
   unfolding landau-o.big.uminus-in-iff by (intro landau-o.big.mult-right π-ϑ-bigo)
  also have (λx::real. x / (ln x)^2 * ln x) ∈ o(λx. x / ln x * ln x)
    by real-asymp
  also have (λx. x / ln x * ln x) ∈ Θ(λx. π x * ln x)
    by (intro asymp-equiv-imp-bigtheta asymp-equiv-intros asymp-equiv-symI[OF
assms])
  finally show (λx. ϑ x − π x * ln x) ∈ o(λx. π x * ln x) .
 qed
 also have ... ∼[at-top] (λx. x / ln x * ln x)
   by (intro asymp-equiv-intros assms)
 also have ... ∼[at-top] (λx. x)
   by real-asymp
 finally show ?thesis .
qed

lemma PNT5-imp-PNT4:
 assumes ψ ∼[at-top] (λx. x)
 shows   ϑ ∼[at-top] (λx. x)
proof −
 define r where r = (λx. ϑ x − ψ x)
 have (λx. ψ x − ϑ x) ∈ O(λx. ln x * sqrt x)
   by (fact ψ-minus-ϑ-bigo)
 also have (λx. ψ x − ϑ x) = (λx. −r x)
   by (simp add: r-def)
 finally have r ∈ O(λx. ln x * sqrt x)
   by simp
```

```
  also have (λx::real. ln x * sqrt x) ∈ o(λx. x)
    by real-asymp
  finally have r: r ∈ o(λx. x) .

  have (λx. ψ x + r x) ∼[at-top] (λx. x)
    using assms r by (subst asymp-equiv-add-right) auto
  thus ?thesis by (simp add: r-def)
qed
```

3.7 The asymptotic form of Mertens' First Theorem

Mertens' first theorem states that $\mathfrak{M}(x) - \ln x$ is bounded, i. e. $\mathfrak{M}(x) = \ln x + O(1)$.

With some work, one can also show some absolute bounds for $|\mathfrak{M}(x) - \ln x|$, and we will, in fact, do this later. However, this asymptotic form is somewhat easier to obtain and it is (as we shall see) enough to prove the Prime Number Theorem, so we prove the weak form here first for the sake of a smoother presentation.

First of all, we need a very weak version of Stirling's formula for the logarithm of the factorial, namely:

$$\ln(\lfloor x \rfloor !) = \sum_{n \leq x} \ln x = x \ln x + O(x)$$

We show this using summation by parts.

```
lemma stirling-weak:
  assumes x: x ≥ 1
  shows   sum-upto ln x ∈ {x * ln x - x - ln x + 1 .. x * ln x}
proof (cases x = 1)
  case True
  have {0<..Suc 0} = {1} by auto
  with True show ?thesis by (simp add: sum-upto-altdef)
next
  case False
  with assms have x: x > 1 by simp
  have ((λt. sum-upto (λ-. 1) t * (1 / t)) has-integral
        sum-upto (λ-. 1) x * ln x - sum-upto (λ-. 1) 1 * ln 1 -
        (∑ n∈real -` {1<..x}. 1 * ln (real n))) {1..x} using x
    by (intro partial-summation-strong[of {}])
       (auto simp flip: has-real-derivative-iff-has-vector-derivative
             intro!: derivative-eq-intros continuous-intros)
  hence ((λt. real (nat ⌊t⌋) / t) has-integral
         real (nat ⌊x⌋) * ln x - (∑ n∈real -` {1<..x}. ln (real n))) {1..x}
    by (simp add: sum-upto-altdef)
  also have (∑ n∈real -` {1<..x}. ln (real n)) = sum-upto ln x unfolding
sum-upto-def
    by (intro sum.mono-neutral-left)
```

```
    (auto intro!: finite-subset[OF - finite-vimage-real-of-nat-greaterThanAtMost[of
0 x]])
  finally have *: ((λt. real (nat ⌊t⌋)) / t) has-integral ⌊x⌋ * ln x − sum-upto ln x)
{1..x}
    using x by simp

  have 0 ≤ real-of-int ⌊x⌋ * ln x − sum-upto (λn. ln (real n)) x
    using * by (rule has-integral-nonneg) auto
  also have ... ≤ x * ln x − sum-upto ln x
    using x by (intro diff-mono mult-mono) auto
  finally have upper: sum-upto ln x ≤ x * ln x by simp

  have (x − 1) * ln x − x + 1 ≤ ⌊x⌋ * ln x − x + 1
    using x by (intro diff-mono mult-mono add-mono) auto
  also have ((λt. 1) has-integral (x − 1)) {1..x}
    using has-integral-const-real[of 1::real 1 x] x by simp
  from * and this have ⌊x⌋ * ln x − sum-upto ln x ≤ x − 1
    by (rule has-integral-le) auto
  hence ⌊x⌋ * ln x − x + 1 ≤ sum-upto ln x
    by simp
  finally have sum-upto ln x ≥ x * ln x − x − ln x + 1
    by (simp add: algebra-simps)
  with upper show ?thesis by simp
qed

lemma stirling-weak-bigo: (λx::real. sum-upto ln x − x * ln x) ∈ O(λx. x)
proof −
  have (λx. sum-upto ln x − x * ln x) ∈ O(λx. −(sum-upto ln x − x * ln x))
    by (subst landau-o.big.uminus) auto
  also have (λx. −(sum-upto ln x − x * ln x)) ∈ O(λx. x + ln x − 1)
   proof (intro le-imp-bigo-real[of 2] eventually-mono[OF eventually-ge-at-top[of
1]], goal-cases)
    case (2 x)
    thus ?case using stirling-weak[of x] by (auto simp: algebra-simps)
  next
    case (3 x)
    thus ?case using stirling-weak[of x] by (auto simp: algebra-simps)
  qed auto
  also have (λx. x + ln x − 1) ∈ O(λx::real. x) by real-asymp
  finally show ?thesis .
qed

lemma floor-floor-div-eq:
  fixes x :: real and d :: nat
  assumes x ≥ 0
  shows   ⌊nat ⌊x⌋ / real d⌋ = ⌊x / real d⌋
proof −
  have ⌊nat ⌊x⌋ / real-of-int (int d)⌋ = ⌊x / real-of-int (int d)⌋ using assms
    by (subst (1 2) floor-divide-real-eq-div) auto
```

thus *?thesis* **by** *simp*
qed

The key to showing Mertens' first theorem is the function

$$h(x) := \sum_{n \leq x} \frac{\Lambda(d)}{d}$$

where Λ is the Mangoldt function, which is equal to $\ln p$ for any prime power p^k and 0 otherwise. As we shall see, $h(x)$ is a good approximation for $\mathfrak{M}(x)$, as the difference between them is bounded by a constant.

lemma *sum-upto-mangoldt-over-id-minus-phi-bounded*:
(λx. *sum-upto* (λd. *mangoldt d / real d*) *x* − 𝔐 *x*) ∈ *O*(λ-. *1*)
proof −
define *f* **where** *f* = (λd. *mangoldt d / real d*)
define *C* **where** *C* = (∑ *p*. *ln* (*real* (*p* + *1*)) ∗ (*1 / real* (*p* ∗ (*p* − *1*))))
have *summable*: *summable* (λp::*nat*. *ln* (*p* + *1*) ∗ (*1 / real* (*p* ∗ (*p* − *1*))))
proof (*rule summable-comparison-test-bigo*)
show *summable* (λp. *norm* (*p powr* (−*3/2*)))
by (*simp add*: *summable-real-powr-iff*)
qed *real-asymp*

have *diff-bound*: *sum-upto f x* − 𝔐 *x* ∈ {*0*..*C*} **if** *x*: *x* ≥ *4* **for** *x*
proof −
define *S* **where** *S* = {(*p*, *i*). *prime p* ∧ *0* < *i* ∧ *real* (*p* ^ *i*) ≤ *x*}
define *S'* **where** *S'* = (*SIGMA p*:{*2*..*nat* ⌊*root 2 x*⌋}. {*2*..*nat* ⌊*log 2 x*⌋})
have *S* ⊆ {..*nat* ⌊*x*⌋} × {..*nat* ⌊*log 2 x*⌋} **unfolding** *S-def*
using *x primepows-le-subset*[*of x 1*] **by** (*auto simp*: *Suc-le-eq*)
hence *finite S* **by** (*rule finite-subset*) *auto*
note *fin* = *finite-subset*[*OF - this*, *unfolded S-def*]

have *sum-upto f x* = (∑ (*p*, *i*)∈*S*. *ln* (*real p*) / *real* (*p* ^ *i*)) **unfolding** *S-def*
by (*intro sum-upto-primepows*) (*auto simp*: *f-def mangoldt-non-primepow*)
also have *S* = {*p*. *prime p* ∧ *p* ≤ *x*} × {*1*} ∪ {(*p*, *i*). *prime p* ∧ *1* < *i* ∧ *real* (*p* ^ *i*) ≤ *x*}
by (*auto simp*: *S-def not-less le-Suc-eq not-le intro*!: *Suc-lessI*)
also have (∑ (*p*,*i*)∈.... *ln* (*real p*) / *real* (*p* ^ *i*)) =
(∑ (*p*, *i*) ∈ {*p*. *prime p* ∧ *of-nat p* ≤ *x*} × {*1*}. *ln* (*real p*) / *real* (*p* ^ *i*)) +
(∑ (*p*, *i*) | *prime p* ∧ *real* (*p* ^ *i*) ≤ *x* ∧ *i* > *1*. *ln* (*real p*) / *real* (*p* ^ *i*))
(**is** - = *?S1* + *?S2*)
by (*subst sum.union-disjoint*[*OF fin fin*]) (*auto simp*: *conj-commute case-prod-unfold*)
also have *?S1* = 𝔐 *x*
by (*subst sum.cartesian-product* [*symmetric*]) (*auto simp*: *primes-M-def prime-sum-upto-def*)
finally have *eq*: *sum-upto f x* − 𝔐 *x* = *?S2* **by** *simp*
have *?S2* ≤ (∑ (*p*, *i*)∈*S'*. *ln* (*real p*) / *real* (*p* ^ *i*))
using *primepows-le-subset*[*of x 2*] *x* **unfolding** *case-prod-unfold of-nat-power*
by (*intro sum-mono2 divide-nonneg-pos zero-less-power*)

```
    (auto simp: eval-nat-numeral Suc-le-eq S'-def subset-iff dest: prime-gt-1-nat)+
  also have ... = (∑ p=2..nat ⌊sqrt x⌋. ln p * (∑ i∈{2..nat ⌊log 2 x⌋}. (1 /
real p) ^ i))
    by (simp add: S'-def sum.Sigma case-prod-unfold
          sum-distrib-left sqrt-def field-simps)
  also have ... ≤ (∑ p=2..nat ⌊sqrt x⌋. ln p * (1 / (p * (p - 1))))
    unfolding sum-upto-def
  proof (intro sum-mono, goal-cases)
    case (1 p)
    from x have nat ⌊log 2 x⌋ ≥ 2
      by (auto simp: le-nat-iff' le-log-iff)
    hence (∑ i∈{2..nat ⌊log 2 x⌋}. (1 / real p) ^ i) =
          ((1 / p)^2 - (1 / p) ^ nat ⌊log 2 x⌋ / p) / (1 - 1 / p) using 1
    by (subst sum-gp) (auto dest!: prime-gt-1-nat simp: field-simps power2-eq-square)
    also have ... ≤ ((1 / p) ^ 2 - 0) / (1 - 1 / p)
      using 1 by (intro divide-right-mono diff-mono power-mono)
            (auto simp: field-simps dest: prime-gt-0-nat)
    also have ... = 1 / (p * (p - 1))
      by (auto simp: divide-simps power2-eq-square dest: prime-gt-0-nat)
    finally show ?case
      using 1 by (intro mult-left-mono) (auto dest: prime-gt-0-nat)
  qed
  also have ... ≤ (∑ p=2..nat ⌊sqrt x⌋. ln (p + 1) * (1 / (p * (p - 1))))
    by (intro sum-mono mult-mono) auto
  also have ... ≤ C unfolding C-def
    by (intro sum-le-suminf summable) auto
  finally have ?S2 ≤ C by simp
  moreover have ?S2 ≥ 0 by (intro sum-nonneg) (auto dest: prime-gt-0-nat)
  ultimately show ?thesis using eq by simp
qed

from diff-bound[of 4] have C ≥ 0 by auto
with diff-bound show (λx. sum-upto f x - 𝔐 x) ∈ O(λ-. 1)
  by (intro le-imp-bigo-real[of C] eventually-mono[OF eventually-ge-at-top[of 4]])
auto
qed
```

Next, we show that our $h(x)$ itself is close to $\ln x$, i. e.:

$$\sum_{n \leq x} \frac{\Lambda(d)}{d} = \ln x + O(1)$$

```
lemma sum-upto-mangoldt-over-id-asymptotics:
  (λx. sum-upto (λd. mangoldt d / real d) x - ln x) ∈ O(λ-. 1)
proof -
  define r where r = (λn::real. sum-upto (λd. mangoldt d * (n / d - real-of-int
⌊n / d⌋)) n)
  have r: r ∈ O(ψ)
  proof (intro landau-o.bigI[of 1] eventually-mono[OF eventually-ge-at-top[of 0]])
```

```
    fix x :: real assume x: x ≥ 0
    have eq: {1..nat ⌊x⌋} = {0<..nat ⌊x⌋} by auto
    hence r x ≥ 0 unfolding r-def sum-upto-def
      by (intro sum-nonneg mult-nonneg-nonneg mangoldt-nonneg)
        (auto simp: floor-le-iff)
    moreover have x / real d ≤ 1 + real-of-int ⌊x / real d⌋ for d by linarith
    hence r x ≤ sum-upto (λd. mangoldt d * 1) x unfolding sum-upto-altdef eq
r-def using x
      by (intro sum-mono mult-mono mangoldt-nonneg)
        (auto simp: less-imp-le[OF frac-lt-1] algebra-simps)
    ultimately show norm (r x) ≤ 1 * norm (ψ x) by (simp add: ψ-def)
  qed auto
  also have ψ ∈ O(λx. x) by (fact ψ-bigo)
  finally have r: r ∈ O(λx. x) .

  define r' where r' = (λx::real. sum-upto ln x - x * ln x)
  have r'-bigo: r' ∈ O(λx. x)
    using stirling-weak-bigo unfolding r'-def .
  have ln-fact: ln (fact n) = (∑ d=1..n. ln d) for n
    by (induction n) (simp-all add: ln-mult)
  hence r': sum-upto ln n = n * ln n + r' n for n :: real
    unfolding r'-def sum-upto-altdef by (auto intro!: sum.cong)

  have eventually (λn. sum-upto (λd. mangoldt d / d) n - ln n = r' n / n + r n
/ n) at-top
    using eventually-gt-at-top
  proof eventually-elim
    fix x :: real assume x: x > 0
    have sum-upto ln x = sum-upto (λn. mangoldt n * real (nat ⌊x / n⌋)) x
      unfolding sum-upto-ln-conv-sum-upto-mangoldt ..
    also have ... = sum-upto (λd. mangoldt d * (x / d)) x - r x
      unfolding sum-upto-def by (simp add: algebra-simps sum-subtractf r-def
sum-upto-def)
    also have sum-upto (λd. mangoldt d * (x / d)) x = x * sum-upto (λd. mangoldt
d / d) x
      unfolding sum-upto-def by (subst sum-distrib-left) (simp add: field-simps)
    finally have x * sum-upto (λd. mangoldt d / real d) x = r' x + r x + x * ln x
      by (simp add: r' algebra-simps)
    thus sum-upto (λd. mangoldt d / d) x - ln x = r' x / x + r x / x
      using x by (simp add: field-simps)
  qed
  hence (λx. sum-upto (λd. mangoldt d / d) x - ln x) ∈ Θ(λx. r' x / x + r x /
x)
    by (rule bigthetaI-cong)
  also have (λx. r' x / x + r x / x) ∈ O(λ-. 1)
    by (intro sum-in-bigo) (insert r r'-bigo, auto simp: landau-divide-simps)
  finally show ?thesis .
qed
```

Combining these two gives us Mertens' first theorem.

theorem *mertens-bounded*: $(\lambda x.\ \mathfrak{M}\ x - ln\ x) \in O(\lambda\text{-}.\ 1)$
proof −
  define f **where** $f = sum\text{-}upto\ (\lambda d.\ mangoldt\ d\ /\ d)$
  have $(\lambda x.\ (f\ x - ln\ x) - (f\ x - \mathfrak{M}\ x)) \in O(\lambda\text{-}.\ 1)$
    using *sum-upto-mangoldt-over-id-asymptotics*
        sum-upto-mangoldt-over-id-minus-phi-bounded
    unfolding *f-def* **by** (*rule sum-in-bigo*)
  thus *?thesis* **by** *simp*
qed

lemma *primes-M-bigo*: $\mathfrak{M} \in O(\lambda x.\ ln\ x)$
proof −
  have $(\lambda x.\ \mathfrak{M}\ x - ln\ x) \in O(\lambda\text{-}.\ 1)$
    by (*rule mertens-bounded*)
  also have $(\lambda\text{-}::real.\ 1) \in O(\lambda x.\ ln\ x)$
    by *real-asymp*
  finally have $(\lambda x.\ \mathfrak{M}\ x - ln\ x + ln\ x) \in O(\lambda x.\ ln\ x)$
    by (*rule sum-in-bigo*) *auto*
  thus *?thesis* **by** *simp*
qed

end

4 The Prime Number Theorem

theory *Prime-Number-Theorem*
imports
  Newman-Ingham-Tauberian
  Prime-Counting-Functions
begin

4.1 Constructing Newman's function

Starting from Mertens' first theorem, i. e. $\mathfrak{M}(x) = \ln x + O(1)$, we now want to derive that $\mathfrak{M}(x) = \ln x + c + o(1)$. This result is considerably stronger and it implies the Prime Number Theorem quite directly.

In order to do this, we define the Dirichlet series

$$f(s) = \sum_{n=1}^{\infty} \frac{\mathfrak{M}(n)}{n^s}\ .$$

We will prove that this series extends meromorphically to $\mathfrak{R}(s) \geq 1$ and apply Ingham's theorem to it (after we subtracted its pole at $s = 1$).

definition *fds-newman* **where**
  fds-newman = *fds* $(\lambda n.\ complex\text{-}of\text{-}real\ (\mathfrak{M}\ n))$

lemma *fds-nth-newman*:
fds-nth fds-newman n = of-real ($\mathfrak{M}$ *n*)
by (*simp add*: *fds-newman-def fds-nth-fds*)

lemma *norm-fds-nth-newman*:
norm (*fds-nth fds-newman n*) = $\mathfrak{M}$ *n*
unfolding *fds-nth-newman norm-of-real*
by (*intro abs-of-nonneg sum-nonneg divide-nonneg-pos*) (*auto dest*: *prime-ge-1-nat*)

The Dirichlet series $f(s) + \zeta'(s)$ has the coefficients $\mathfrak{M}(n) - \ln n$, so by Mertens' first theorem, $f(s) + \zeta'(s)$ has bounded coefficients.

lemma *bounded-coeffs-newman-minus-deriv-zeta*:
defines *f* ≡ *fds-newman + fds-deriv fds-zeta*
shows *Bseq* (λ*n*. *fds-nth f n*)
proof −
have (λ*n*. $\mathfrak{M}$ (*real n*) − *ln* (*real n*)) ∈ *O*(λ-. *1*)
using *mertens-bounded* **by** (*rule landau-o.big.compose*) *real-asymp*
from *natfun-bigo-1E*[*OF this*, *of 1*]
obtain *c* **where** *c*: *c* ≥ *1* ⋀*n*. |$\mathfrak{M}$ (*real n*) − *ln* (*real n*)| ≤ *c* **by** *auto*

show *?thesis*
proof (*intro BseqI*[*of c*] *allI*)
fix *n* :: *nat*
show *norm* (*fds-nth f n*) ≤ *c*
proof (*cases n = 0*)
case *False*
hence *fds-nth f n = of-real* ($\mathfrak{M}$ *n* − *ln n*)
by (*simp add*: *f-def fds-nth-newman fds-nth-deriv fds-nth-zeta scaleR-conv-of-real*)
also from ‹*n* ≠ *0*› **have** *norm* … ≤ *c*
using *c*(*2*)[*of n*] **by** (*simp add*: *in-Reals-norm*)
finally show *?thesis* **.**
qed (*insert c*, *auto*)
qed (*insert c*, *auto*)
qed

A Dirichlet series with bounded coefficients converges for all s with $\Re(s) > 1$ and so does $\zeta'(s)$, so we can conclude that $f(s)$ does as well.

lemma *abs-conv-abscissa-newman*: *abs-conv-abscissa fds-newman* ≤ *1*
and *conv-abscissa-newman*: *conv-abscissa fds-newman* ≤ *1*
proof −
define *f* **where** *f* = *fds-newman + fds-deriv fds-zeta*
have *abs-conv-abscissa f* ≤ *1*
using *bounded-coeffs-newman-minus-deriv-zeta* **unfolding** *f-def*
by (*rule bounded-coeffs-imp-abs-conv-abscissa-le-1*)
hence *abs-conv-abscissa* (*f* − *fds-deriv fds-zeta*) ≤ *1*
by (*intro abs-conv-abscissa-diff-leI*) (*auto simp*: *abs-conv-abscissa-deriv*)
also have *f* − *fds-deriv fds-zeta* = *fds-newman* **by** (*simp add*: *f-def*)
finally show *abs-conv-abscissa fds-newman* ≤ *1* **.**
from *conv-le-abs-conv-abscissa* **and** *this* **show** *conv-abscissa fds-newman* ≤ *1*

by (*rule order.trans*)
qed

We now change the order of summation to obtain an alternative form of $f(s)$ in terms of a sum of Hurwitz ζ functions.

lemma *eval-fds-newman-conv-infsetsum*:
assumes *s*: *Re s > 1*
shows *eval-fds fds-newman s = (∑ₐp | prime p. (ln (real p) / real p) * hurwitz-zeta p s)*
*(λp. ln (real p) / real p * hurwitz-zeta p s) abs-summable-on {p. prime p}*
proof −
from *s* **have** *conv*: *fds-abs-converges fds-newman s*
by (*intro fds-abs-converges le-less-trans*[*OF abs-conv-abscissa-newman*]) *auto*
define *f* **where** *f = (λn p. ln (real p) / real p / of-nat n powr s)*

have *eq*: *(∑ₐn∈{p..}. f n p) = ln (real p) / real p * hurwitz-zeta p s* **if** *prime p* **for** *p*
proof −
have *(∑ₐn∈{p..}. f n p) = (∑ₐx∈{p..}. (ln (real p) / of-nat p) * (1 / of-nat x powr s))*
by (*simp add*: *f-def*)
also have *... = (ln (real p) / of-nat p) * (∑ₐx∈{p..}. 1 / of-nat x powr s)*
using *abs-summable-hurwitz-zeta*[*of s 0 p*] *that s*
by (*intro infsetsum-cmult-right*) (*auto dest*: *prime-gt-0-nat*)
also have *(∑ₐx∈{p..}. 1 / of-nat x powr s) = hurwitz-zeta p s*
using *s that* **by** (*subst hurwitz-zeta-nat-conv-infsetsum*(*2*))
(*auto dest*: *prime-gt-0-nat simp*: *field-simps powr-minus*)
finally show *?thesis* **.**
qed

have *norm-f*: *norm (f n p) = ln p / p / n powr Re s* **if** *prime p* **for** *n p* :: *nat*
by (*auto simp*: *f-def norm-divide norm-mult norm-powr-real-powr*)
from *conv* **have** *(λn. norm (fds-nth fds-newman n / n powr s)) abs-summable-on UNIV*
by (*intro abs-summable-on-normI*) (*simp add*: *fds-abs-converges-altdef'*)
also have *(λn. norm (fds-nth fds-newman n / n powr s)) =*
(λn. ∑ p | prime p ∧ p ≤ n. norm (f n p))
by (*auto simp*: *norm-divide norm-fds-nth-newman sum-divide-distrib primes-M-def*
prime-sum-upto-def norm-mult norm-f norm-powr-real-powr intro!: *sum.cong*)
finally have *summable1*: *(λ(n,p). f n p) abs-summable-on (SIGMA n:UNIV. {p. prime p ∧ p ≤ n})*
using *conv* **by** (*subst abs-summable-on-Sigma-iff*) *auto*
also have *?this ⟷ (λ(p,n). f n p) abs-summable-on*
(λ(n,p). (p,n)) ‘ (SIGMA n:UNIV. {p. prime p ∧ p ≤ n})
by (*subst abs-summable-on-reindex-iff* [*symmetric*]) (*auto simp*: *case-prod-unfold inj-on-def*)
also have *(λ(n,p). (p,n)) ‘ (SIGMA n:UNIV. {p. prime p ∧ p ≤ n}) =*
(SIGMA p:{p. prime p}. {p..}) **by** *auto*

finally have *summable2*: $(\lambda(p,n).\ f\ n\ p)$ *abs-summable-on* ... .
from *abs-summable-on-Sigma-project1* '[*OF this*]
have $(\lambda p.\ \sum_a n{\in}\{p..\}.\ f\ n\ p)$ *abs-summable-on* $\{p.\ prime\ p\}$ **by** *auto*
also have *?this* $\longleftrightarrow$ $(\lambda p.\ ln\ (real\ p)\ /\ real\ p * hurwitz\text{-}zeta\ p\ s)$ *abs-summable-on* $\{p.\ prime\ p\}$
by (*intro abs-summable-on-cong eq*) *auto*
finally show ... .

have *eval-fds fds-newman s* =
$(\sum_a n.\ \sum p \mid prime\ p \wedge p \leq n.\ ln\ (real\ p)\ /\ real\ p\ /\ of\text{-}nat\ n\ powr\ s)$
using *conv* **by** (*simp add*: *eval-fds-altdef fds-nth-newman sum-divide-distrib primes-M-def prime-sum-upto-def*)
also have ... = $(\sum_a n.\ \sum_a p \mid prime\ p \wedge p \leq n.\ f\ n\ p)$
unfolding *f-def* **by** (*subst infsetsum-finite*) *auto*
also have ... = $(\sum_a (n,\ p) \in (SIGMA\ n{:}UNIV.\ \{p.\ prime\ p \wedge p \leq n\}).\ f\ n\ p)$
using *summable1* **by** (*subst infsetsum-Sigma*) *auto*
also have ... = $(\sum_a (p,\ n) \in (\lambda(n,p).\ (p,\ n))\ `\ (SIGMA\ n{:}UNIV.\ \{p.\ prime\ p \wedge p \leq n\}).\ f\ n\ p)$
by (*subst infsetsum-reindex*) (*auto simp*: *case-prod-unfold inj-on-def*)
also have $(\lambda(n,p).\ (p,\ n))\ `\ (SIGMA\ n{:}UNIV.\ \{p.\ prime\ p \wedge p \leq n\}) =$
$(SIGMA\ p{:}\{p.\ prime\ p\}.\ \{p..\})$ **by** *auto*
also have $(\sum_a (p,n){\in}....\ f\ n\ p) = (\sum_a p \mid prime\ p.\ \sum_a n{\in}\{p..\}.\ f\ n\ p)$
using *summable2* **by** (*subst infsetsum-Sigma*) *auto*
also have $(\sum_a p \mid prime\ p.\ \sum_a n{\in}\{p..\}.\ f\ n\ p) =$
$(\sum_a p \mid prime\ p.\ ln\ (real\ p)\ /\ real\ p * hurwitz\text{-}zeta\ p\ s)$
by (*intro infsetsum-cong eq*) *auto*
finally show *eval-fds fds-newman s* =
$(\sum_a p \mid prime\ p.\ (ln\ (real\ p)\ /\ real\ p) * hurwitz\text{-}zeta\ p\ s)$.
qed

We now define a meromorphic continuation of $f(s)$ on $\Re(s) > \frac{1}{2}$.
To construct $f(s)$, we express it as

$$f(s) = \frac{1}{z-1}\left(\bar{f}(s) - \frac{\zeta'(s)}{\zeta(s)}\right)\ ,$$

where $\bar{f}(s)$ (which we shall call *pre-newman*) is a function that is analytic on $\Re(s) > \frac{1}{2}$, which can be shown fairly easily using the Weierstraß M test. $\zeta'(s)/\zeta(s)$ is meromorphic except for a single pole at $s = 1$ and one k-th order pole for any k-th order zero of ζ, but for the Prime Number Theorem, we are only concerned with the area $\Re(s) \geq 1$, where ζ does not have any zeros.

Taken together, this means that $f(s)$ is analytic for $\Re(s) \geq 1$ except for a double pole at $s = 1$, which we will take care of later.

context
fixes $A :: nat \Rightarrow complex \Rightarrow complex$ **and** $B :: nat \Rightarrow complex \Rightarrow complex$
defines $A \equiv (\lambda p\ s.\ (s - 1) * pre\text{-}zeta\ (real\ p)\ s -$
$of\text{-}nat\ p\ /\ (of\text{-}nat\ p\ powr\ s * (of\text{-}nat\ p\ powr\ s - 1)))$

defines $B \equiv (\lambda p\ s.\ of\text{-}real\ (ln\ (real\ p))\ /\ of\text{-}nat\ p * A\ p\ s)$
begin

definition *pre-newman* :: *complex* ⇒ *complex* **where**
pre-newman s = $(\sum p.\ if\ prime\ p\ then\ B\ p\ s\ else\ 0)$

definition *newman* **where** *newman s* = *1 / (s − 1) * (pre-newman s − deriv zeta s / zeta s)*

The sum used in the definition of *pre-newman* converges uniformly on any disc within the half-space with $\Re(s) > \frac{1}{2}$ by the Weierstraß M test.

lemma *uniform-limit-pre-newman*:
assumes *r*: *r ≥ 0 Re s − r > 1 / 2*
shows *uniform-limit* (*cball s r*)
($\lambda n\ s.\ \sum p{<}n.\ if\ prime\ p\ then\ B\ p\ s\ else\ 0$) *pre-newman at-top*
proof −
from *r* **have** *Re*: *Re z > 1 / 2* **if** *dist s z ≤ r* **for** *z*
using *abs-Re-le-cmod*[*of s − z*] *r that*
by (*auto simp*: *dist-norm abs-if split*: *if-splits*)

define *x* **where** *x* = *Re s − r* — The lower bound for the real part in the disc
from *r Re* **have** *x > 1 / 2* **by** (*auto simp*: *x-def*)

— The following sequence *M* bounds the summand, and it is obviously $O(n^{-1-\epsilon})$ and therefore summable
define *C* **where** *C* = (*norm s + r + 1*) * (*norm s + r*) / *x*
define *M* **where** *M* = (λ*p*::*nat*. *ln p * (C / p powr (x + 1)) + 1 / (p powr x * (p powr x − 1))))

show *?thesis* **unfolding** *pre-newman-def*
proof (*intro Weierstrass-m-test-ev*[*OF eventually-mono*[*OF eventually-gt-at-top*[*of 1*]]] *ballI*)
show *summable M*
proof (*rule summable-comparison-test-bigo*)
define ε **where** ε = *min* (*2 * x − 1*) *x / 2*
from ‹*x > 1 / 2*› **have** ε: ε > *0 1* + ε < *2 * x 1* + ε < *x + 1*
by (*auto simp*: ε-*def min-def field-simps*)
show *M* ∈ *O*(λ*n*. *n powr* (− *1* − ε)) **unfolding** *M-def distrib-left*
by (*intro sum-in-bigo*) (*use* ε **in** *real-asymp*)+
from ε **show** *summable* (λ*n*. *norm* (*n powr* (− *1* − ε)))
by (*simp add*: *summable-real-powr-iff*)
qed
next
fix *p* :: *nat* **and** *z* **assume** *p*: *p > 1* **and** *z*: *z* ∈ *cball s r*
from *z r Re*[*of z*] **have** *x*: *Re z ≥ x x > 1 / 2* **and** *Re z > 1 / 2*
using *abs-Re-le-cmod*[*of s − z*] **by** (*auto simp*: *x-def algebra-simps dist-norm*)
have *norm-z*: *norm z ≤ norm s + r*
using *z norm-triangle-ineq2*[*of z s*] *r* **by** (*auto simp*: *dist-norm norm-minus-commute*)
from ‹*p > 1*› **and** *x* **and** *r* **have** *M p ≥ 0*

```
    by (auto simp: C-def M-def intro!: mult-nonneg-nonneg add-nonneg-nonneg
divide-nonneg-pos)

  have bound: norm ((z - 1) * pre-zeta p z) ≤
                norm (z - 1) * (norm z / (Re z * p powr Re z))
    using pre-zeta-bound'[of z p] p ‹Re z > 1 / 2›
    unfolding norm-mult by (intro mult-mono pre-zeta-bound) auto

  have norm (B p z) = ln p / p * norm (A p z)
   unfolding B-def using ‹p > 1› by (simp add: B-def norm-mult norm-divide)
  also have ... ≤ ln p / p * (norm (z - 1) * norm z / Re z / p powr Re z +
                                p / (p powr Re z * (p powr Re z - 1)))
    unfolding A-def using ‹p > 1› and ‹Re z > 1 / 2› and bound
   by (intro mult-left-mono order.trans[OF norm-triangle-ineq4 add-mono] mult-left-mono)
      (auto simp: norm-divide norm-mult norm-powr-real-powr
            intro!: divide-left-mono order.trans[OF - norm-triangle-ineq2])
  also have ... = ln p * (norm (z - 1) * norm z / Re z / p powr (Re z + 1)
+
                          1 / (p powr Re z * (p powr Re z - 1)))
    using ‹p > 1› by (simp add: field-simps powr-add powr-minus)
  also have norm (z - 1) * norm z / Re z / p powr (Re z + 1) ≤ C / p powr
(x + 1)
    unfolding C-def using r ‹Re z > 1 / 2› norm-z p x
    by (intro mult-mono frac-le powr-mono order.trans[OF norm-triangle-ineq4])
auto
  also have 1 / (p powr Re z * (p powr Re z - 1)) ≤
              1 / (p powr x * (p powr x - 1)) using ‹p > 1› x
   by (intro divide-left-mono mult-mono powr-mono diff-right-mono mult-pos-pos)
      (auto simp: ge-one-powr-ge-zero)
  finally have norm (B p z) ≤ M p
    using ‹p > 1› by (simp add: mult-left-mono M-def)
  with ‹M p ≥ 0› show norm (if prime p then B p z else 0) ≤ M p by simp
 qed
qed

lemma sums-pre-newman: Re s > 1 / 2 ⟹ (λp. if prime p then B p s else 0)
sums pre-newman s
  using tendsto-uniform-limitI[OF uniform-limit-pre-newman[of 0 s]] by (auto
simp: sums-def)

lemma analytic-pre-newman [THEN analytic-on-subset, analytic-intros]:
 pre-newman analytic-on {s. Re s > 1 / 2}
proof -
 have holo: (λs::complex. if prime p then B p s else 0) holomorphic-on X
   if X ⊆ {s. Re s > 1 / 2} for X and p :: nat using that
   by (cases prime p)
     (auto intro!: holomorphic-intros simp: B-def A-def dest!: prime-gt-1-nat)
 have holo': pre-newman holomorphic-on ball s r if r: r ≥ 0 Re s - r > 1 / 2
for s r
```

```
  proof −
    from r have Re: Re z > 1 / 2 if dist s z ≤ r for z
      using abs-Re-le-cmod[of s − z] r that by (auto simp: dist-norm abs-if split:
if-splits)
    show ?thesis
      by (rule holomorphic-uniform-limit[OF - uniform-limit-pre-newman[of r s]])
       (insert that Re, auto intro!: always-eventually holomorphic-on-imp-continuous-on
                                     holomorphic-intros holo)
  qed
  show ?thesis unfolding analytic-on-def
  proof safe
    fix s assume Re s > 1 / 2
    thus ∃r>0. pre-newman holomorphic-on ball s r
      by (intro exI[of - (Re s − 1 / 2) / 2] conjI holo′) (auto simp: field-simps)
  qed
qed

lemma holomorphic-pre-newman [holomorphic-intros]:
  X ⊆ {s. Re s > 1 / 2} ⟹ pre-newman holomorphic-on X
  using analytic-pre-newman by (rule analytic-imp-holomorphic)

lemma eval-fds-newman:
  assumes s: Re s > 1
  shows   eval-fds fds-newman s = newman s
proof −
  have eq: (ln (real p) / real p) ∗ hurwitz-zeta p s =
            1 / (s − 1) ∗ (ln (real p) / (p powr s − 1) + B p s)
    if p: prime p for p
  proof −
    have (ln (real p) / real p) ∗ hurwitz-zeta p s =
          ln (real p) / real p ∗ (p powr (1 − s) / (s − 1) + pre-zeta p s)
      using s by (auto simp add: hurwitz-zeta-def)
    also have ... = 1 / (s − 1) ∗ (ln (real p) / (p powr s − 1) + B p s)
      using p s by (simp add: divide-simps powr-diff B-def)
                   (auto simp: A-def field-simps dest: prime-gt-1-nat)?
    finally show ?thesis .
  qed

  have (λp. (ln (real p) / real p) ∗ hurwitz-zeta p s) abs-summable-on {p. prime
p}
    using s by (intro eval-fds-newman-conv-infsetsum)
  hence (λp. 1 / (s − 1) ∗ (ln (real p) / (p powr s − 1) + B p s))
          abs-summable-on {p. prime p}
    by (subst (asm) abs-summable-on-cong[OF eq refl]) auto
  hence summable:
    (λp. ln (real p) / (p powr s − 1) + B p s) abs-summable-on {p. prime p}
    using s by (subst (asm) abs-summable-on-cmult-right-iff) auto

  from s have [simp]: s ≠ 1 by auto
```

have *eval-fds fds-newman s* =
(∑ₐ *p* | *prime p*. (*ln* (*real p*) / *real p*) ∗ *hurwitz-zeta p s*)
using *s* **by** (*rule eval-fds-newman-conv-infsetsum*)
also have ... = (∑ₐ *p* | *prime p*. *1* / (*s* − *1*) ∗ (*ln* (*real p*) / (*p powr s* − *1*) + *B p s*))
by (*intro infsetsum-cong eq*) *auto*
also have ... = *1* / (*s* − *1*) ∗ (∑ₐ *p* | *prime p*. *ln* (*real p*) / (*p powr s* − *1*) + *B p s*)
(**is** - = - ∗ *?S*) **by** (*rule infsetsum-cmult-right*[*OF summable*])
also have *?S* = (∑ *p*. *if prime p then*
ln (*real p*) / (*p powr s* − *1*) + *B p s else 0*)
by (*subst infsetsum-nat*[*OF summable*]) *auto*
also have ... = (∑ *p*. (*if prime p then ln* (*real p*) / (*p powr s* − *1*) *else 0*) +
(*if prime p then B p s else 0*))
by (*intro suminf-cong*) *auto*
also have ... = *pre-newman s* − *deriv zeta s* / *zeta s*
using *sums-pre-newman*[*of s*] *sums-logderiv-zeta*[*of s*] *s*
by (*subst suminf-add* [*symmetric*]) (*auto simp*: *sums-iff*)
finally show *?thesis* **by** (*simp add*: *newman-def*)
qed

end

Next, we shall attempt to get rid of the pole by subtracting suitable multiples of $\zeta(s)$ and $\zeta'(s)$. To this end, we shall first prove the following alternative definition of $\zeta'(s)$:

lemma *deriv-zeta-eq′*:
assumes *0* < *Re s s* ≠ *1*
shows *deriv zeta s* = *deriv* (λ*z*. *pre-zeta 1 z* ∗ (*z* − *1*)) *s* / (*s* − *1*) −
(*pre-zeta 1 s* ∗ (*s* − *1*) + *1*) / (*s* − *1*)²
(**is** - = *?rhs*)
proof (*rule DERIV-imp-deriv*)
have [*derivative-intros*]: (*pre-zeta 1 has-field-derivative deriv* (*pre-zeta 1*) *s*) (*at s*)
by (*intro holomorphic-derivI*[*of - UNIV*] *holomorphic-intros*) *auto*
have ∗: *deriv* (λ*z*. *pre-zeta 1 z* ∗ (*z* − *1*)) *s* = *deriv* (*pre-zeta 1*) *s* ∗ (*s* − *1*) + *pre-zeta 1 s*
by (*subst deriv-mult*)
(*auto intro*!: *holomorphic-on-imp-differentiable-at*[*of - UNIV*] *holomorphic-intros*)
hence ((λ*s*. *pre-zeta 1 s* + *1* / (*s* − *1*)) *has-field-derivative*
deriv (*pre-zeta 1*) *s* − *1* / ((*s* − *1*) ∗ (*s* − *1*))) (*at s*)
using *assms* **by** (*auto intro*!: *derivative-eq-intros*)
also have *deriv* (*pre-zeta 1*) *s* − *1* / ((*s* − *1*) ∗ (*s* − *1*)) = *?rhs*
using ∗ *assms* **by** (*simp add*: *divide-simps power2-eq-square*, *simp add*: *field-simps*)
also have ((λ*s*. *pre-zeta 1 s* + *1* / (*s* − *1*)) *has-field-derivative ?rhs*) (*at s*) ⟷
(*zeta has-field-derivative ?rhs*) (*at s*)
using *assms*
by (*intro has-field-derivative-cong-ev eventually-mono*[*OF t1-space-nhds*[*of - 1*]])
(*auto simp*: *zeta-def hurwitz-zeta-def*)

```
  finally show ... .
qed
```

From this, it follows that $(s-1)\zeta'(s) - \zeta'(s)/\zeta(s)$ is analytic for $\Re(s) \geq 1$:

```
lemma analytic-zeta-derivdiff:
  obtains a where
    (λz. if z = 1 then a else (z − 1) ∗ deriv zeta z − deriv zeta z / zeta z)
        analytic-on {s. Re s ≥ 1}
proof
  have neq: pre-zeta 1 z ∗ (z − 1) + 1 ≠ 0 if Re z ≥ 1 for z
    using zeta-Re-ge-1-nonzero[of z] that
    by (cases z = 1) (auto simp: zeta-def hurwitz-zeta-def divide-simps)
  let ?g = λz. (1 − inverse (pre-zeta 1 z ∗ (z − 1) + 1)) ∗ ((z − 1) ∗
            deriv ((λu. pre-zeta 1 u ∗ (u − 1))) z − (pre-zeta 1 z ∗ (z − 1) + 1))
  show (λz. if z = 1 then deriv ?g 1 else (z − 1) ∗ deriv zeta z − deriv zeta z /
zeta z)
        analytic-on {s. Re s ≥ 1} (is ?f analytic-on -)
  proof (rule pole-theorem-analytic-0)
    show ?g analytic-on {s. 1 ≤ Re s} using neq
      by (auto intro!: analytic-intros)
  next
    show ∃ d>0. ∀ w∈ball z d − {1}. ?g w = (w − 1) ∗ ?f w
      if z: z ∈ {s. 1 ≤ Re s} for z
    proof −
      have ∗: isCont (λz. pre-zeta 1 z ∗ (z − 1) + 1) z
        by (auto intro!: continuous-intros)
      obtain e where e > 0 and e: ⋀y. dist z y < e ⟹ pre-zeta (Suc 0) y ∗
(y−1) + 1 ≠ 0
        using continuous-at-avoid [OF ∗ neq[of z]] z by auto
      show ?thesis
      proof (intro exI ballI conjI)
        fix w
        assume w: w ∈ ball z (min e 1) − {1}
        then have Re w > 0
          using complex-Re-le-cmod [of z−w] z by (simp add: dist-norm)
        with w show ?g w = (w − 1) ∗ (if w = 1 then deriv ?g 1 else
                    (w − 1) ∗ deriv zeta w − deriv zeta w / zeta w)
          by (subst (1 2) deriv-zeta-eq',
             simp-all add: zeta-def hurwitz-zeta-def divide-simps e power2-eq-square)
             (simp-all add: algebra-simps)?
      qed (use ‹e > 0› in auto)
    qed
  qed auto
qed
```

Finally, $f(s) + \zeta'(s) + c\zeta(s)$ is analytic.

```
lemma analytic-newman-variant:
  obtains c a where
    (λz. if z = 1 then a else newman z + deriv zeta z + c ∗ zeta z) analytic-on
```

```
{s. Re s ≥ 1}
proof −
  obtain c where
    c: (λz. if z = 1 then c else (z − 1) ∗ deriv zeta z − deriv zeta z / zeta z)
        analytic-on {s. Re s ≥ 1}
    using analytic-zeta-derivdiff by blast
  let ?g = λz. pre-newman z +
          (if z = 1 then c
           else (z − 1) ∗ deriv zeta z −
                deriv zeta z / zeta z) − (c + pre-newman 1) ∗ (pre-zeta 1 z ∗ (z −
1) + 1)
   have (λz. if z = 1 then deriv ?g 1 else newman z + deriv zeta z + (−(c +
pre-newman 1)) ∗ zeta z)
        analytic-on {s. Re s ≥ 1}  (is ?f analytic-on -)
  proof (rule pole-theorem-analytic-0)
    show ?g analytic-on {s. 1 ≤ Re s}
      by (intro c analytic-intros) auto
  next
    show ∃ d>0. ∀ w∈ball z d − {1}. ?g w = (w − 1) ∗ ?f w
      if z ∈ {s. 1 ≤ Re s} for z using that
       by (intro exI[of - 1], simp-all add: newman-def divide-simps zeta-def hur-
witz-zeta-def)
          (auto simp: field-simps)?
  qed auto
  with that show ?thesis by blast
qed
```

4.2 The asymptotic expansion of $\mathfrak{M}$

Our next goal is to show the key result that $\mathfrak{M}(x) = \ln n + c + o(1)$.

As a first step, we invoke Ingham's Tauberian theorem on the function we have just defined and obtain that the sum

$$\sum_{n=1}^{\infty} \frac{\mathfrak{M}(n) - \ln n + c}{n}$$

exists.

```
lemma mertens-summable:
  obtains c :: real where summable (λn. (𝔐 n − ln n + c) / n)
proof −

  from analytic-newman-variant obtain c a where
    analytic: (λz. if z = 1 then a else newman z + deriv zeta z + c ∗ zeta z)
                analytic-on {s. Re s ≥ 1} .
  define f where f = (λz. if z = 1 then a else newman z + deriv zeta z + c ∗
zeta z)
  have analytic: f analytic-on {s. Re s ≥ 1} using analytic by (simp add: f-def)
  define F where F = fds-newman + fds-deriv fds-zeta + fds-const c ∗ fds-zeta
```

```
 note le = conv-abscissa-add-leI conv-abscissa-deriv-le conv-abscissa-newman conv-abscissa-mult-const-left
 note intros = le le[THEN le-less-trans] le[THEN order.trans] fds-converges
 have eval-F: eval-fds F s = f s if s: Re s > 1 for s
 proof −
   have eval-fds F s = eval-fds (fds-newman + fds-deriv fds-zeta) s +
                       eval-fds (fds-const c ∗ fds-zeta) s
     unfolding F-def using s by (subst eval-fds-add) (auto intro!: intros)
   also have ... = f s using s unfolding f-def
     by (subst eval-fds-add)
        (auto intro!: intros simp: eval-fds-newman eval-fds-deriv-zeta eval-fds-mult
eval-fds-zeta)
   finally show ?thesis .
 qed

 have conv: fds-converges F s if Re s ≥ 1 for s
 proof (rule Newman-Ingham-1)
   have (λn. 𝔐 (real n) − ln (real n)) ∈ O(λ-. 1)
     using mertens-bounded by (rule landau-o.big.compose) real-asymp
   from natfun-bigo-1E[OF this, of 1]
     obtain c′ where c′: c′ ≥ 1 ⋀n. |𝔐 (real n) − ln (real n)| ≤ c′ by auto
   have Bseq (fds-nth F)
   proof (intro BseqI allI)
     fix n :: nat
     show norm (fds-nth F n) ≤ (c′ + norm c) unfolding F-def using c′
      by (auto simp: fds-nth-zeta fds-nth-deriv fds-nth-newman scaleR-conv-of-real
in-Reals-norm
               intro!: order.trans[OF norm-triangle-ineq] add-mono)
   qed (insert c′, auto intro: add-pos-nonneg)
   thus fds-nth F ∈ O(λ-. 1) by (simp add: natfun-bigo-iff-Bseq)
 next
   show f analytic-on {s. Re s ≥ 1} by fact
 next
   show eval-fds F s = f s if Re s > 1 for s using that by (rule eval-F)
 qed (insert that, auto simp: F-def intro!: intros)
 from conv[of 1] have summable (λn. fds-nth F n / of-nat n)
   unfolding fds-converges-def by auto
 also have ?this ⟷ summable (λn. (𝔐 n − Ln n + c) / n)
   by (intro summable-cong eventually-mono[OF eventually-gt-at-top[of 0]])
    (auto simp: F-def fds-nth-newman fds-nth-deriv fds-nth-zeta scaleR-conv-of-real
          intro!: sum.cong dest: prime-gt-0-nat)
 finally have summable (λn. (𝔐 n − Re (Ln (of-nat n)) + Re c) / n)
   by (auto dest: summable-Re)
 also have ?this ⟷ summable (λn. (𝔐 n − ln n + Re c) / n)
   by (intro summable-cong eventually-mono[OF eventually-gt-at-top[of 0]]) (auto
intro!: sum.cong)
 finally show ?thesis using that[of Re c] by blast
qed
```

Next, we prove a lemma given by Newman stating that if the sum $\sum a_n/n$

exists and $a_n + \ln n$ is nondecreasing, then a_n must tend to 0. Unfortunately, the proof is rather tedious, but so is the paper version by Newman.

```
lemma sum-goestozero-lemma:
  fixes d::real
  assumes d: |∑ i = M..N. a i / i| < d and le: ⋀n. a n + ln n ≤ a (Suc n) +
ln (Suc n)
      and 0 < M M < N
    shows a M ≤ d * N / (real N − real M) + (real N − real M) / M ∧
          −a N ≤ d * N / (real N − real M) + (real N − real M) / M
proof −
  have 0 ≤ d
    using assms by linarith+
  then have 0 ≤ d * N / (N − M + 1) by simp
  then have le-dN: ⟦0 ≤ x ⟹ x ≤ d * N / (N − M + 1)⟧ ⟹ x ≤ d * N / (N
− M + 1) for x::real
    by linarith
  have le-a-ln: a m + ln m ≤ a n + ln n if n ≥ m for n m
    by (rule transitive-stepwise-le) (use le that in auto)
  have ∗: x ≤ b ∧ y ≤ b if a ≤ b x ≤ a y ≤ a for a b x y::real
    using that by linarith
  show ?thesis
  proof (rule ∗)
    show d * N / (N − M) + ln (N / M) ≤ d * N / (real N − real M) + (real
N − real M) / M
      using ‹0 < M› ‹M < N› ln-le-minus-one [of N / M]
      by (simp add: of-nat-diff) (simp add: divide-simps)
  next
    have a M − ln (N / M) ≤ (d * N) / (N − M + 1)
    proof (rule le-dN)
      assume 0: 0 ≤ a M − ln (N / M)
      have (Suc N − M) * (a M − ln (N / M)) / N = (∑ i = M..N. (a M − ln
(N / M)) / N)
        by simp
      also have ... ≤ (∑ i = M..N. a i / i)
      proof (rule sum-mono)
        fix i
        assume i: i ∈ {M..N}
        with ‹0 < M› have 0 < i by auto
        have (a M − ln (N / M)) / N ≤ (a M − ln (N / M)) / i
       using 0 using i ‹0 < M› by (simp add: frac-le-eq divide-simps mult-left-mono)
        also have a M + ln (real M) ≤ a i + ln (real N)
          by (rule order.trans[OF le-a-ln[of M i]]) (use i assms in auto)
        hence (a M − ln (N / M)) / i ≤ a i / real i
          using assms i by (intro divide-right-mono) (auto simp: ln-div field-simps)
        finally show (a M − ln (N / M)) / real N ≤ a i / real i .
      qed
      finally have ((Suc N) − M) * (a M − ln (N / M)) / N ≤ |∑ i = M..N. a
i / i|
        by simp
```

```
    also have ... ≤ d using d by simp
    finally have ((Suc N) − M) * (a M − ln (N / M)) / N ≤ d .
    then show ?thesis
      using ‹M < N› by (simp add: of-nat-diff field-simps)
  qed
  also have ... ≤ d * N / (N − M)
    using assms(1,4) by (simp add: field-simps)
  finally show a M ≤ d * N / (N − M) + ln (N / M) by simp
 next
  have − a N − ln (N / M) ≤ (d * N) / (N − M + 1)
  proof (rule le-dN)
    assume 0: 0 ≤ − a N − ln (N / M)
    have (∑ i = M..N. a i / i) ≤ (∑ i = M..N. (a N + ln (N / M)) / N)
    proof (rule sum-mono)
      fix i
      assume i: i ∈ {M..N}
      with ‹0 < M› have 0 < i by auto
      have a i + ln (real M) ≤ a N + ln (real N)
        by (rule order.trans[OF - le-a-ln[of i N]]) (use i assms in auto)
      hence a i / i ≤ (a N + ln (N / M)) / i
          using assms(3,4) by (intro divide-right-mono) (auto simp: field-simps
ln-div)
      also have ... ≤ (a N + ln (N / M)) / N
        using i ‹i > 0› 0 by (intro divide-left-mono-neg) auto
      finally show a i / i ≤ (a N + ln (N / M)) / N .
    qed
    also have ... = ((Suc N) − M) * (a N + ln (N / M)) / N
      by simp
    finally have (∑ i = M..N. a i / i) ≤ (real (Suc N) − real M) * (a N + ln
(N / M)) / N
      using ‹M < N› by (simp add: of-nat-diff)
    then have −((real (Suc N) − real M) * (a N + ln (N / M)) / N) ≤ |∑ i
= M..N. a i / i|
      by linarith
    also have ... ≤ d using d by simp
    finally have − ((real (Suc N) − real M) * (a N + ln (N / M)) / N) ≤ d .
    then show ?thesis
      using ‹M < N› by (simp add: of-nat-diff field-simps)
  qed
  also have ... ≤ d * N / real (N − M) by (simp add: field-simps)
    using ‹0 < M› ‹M < N› ‹0 ≤ d› by (simp add: field-simps)
  finally show −a N ≤ d * N / real (N − M) + ln (N / M) by simp
 qed
qed

proposition sum-goestozero-theorem:
 assumes summ: summable (λi. a i / i)
    and le: ⋀n. a n + ln n ≤ a (Suc n) + ln (Suc n)
  shows a ⟶ 0
```

proof (*clarsimp simp*: *lim-sequentially*)
fix *r*::*real*
assume *r* > *0*
have ∗: ∃ *n0*. ∀ *n*≥*n0*. |*a n*| < *ε* **if** *ε*: *0* < *ε ε* < *1* **for** *ε*
proof −
have *0* < (*ε* / *8*)^2 **using** ‹*0* < *ε*› **by** *simp*
then obtain *N0* **where** *N0*: ⋀*m n*. *m* ≥ *N0* ⟹ *norm* (∑ *k*=*m*..*n*. (λ*i*. *a i* / *i*) *k*) < (*ε* / *8*)^2
by (*metis summable-partial-sum-bound summ*)
obtain *N1* **where** *real N1* > *4* / *ε*
using *reals-Archimedean2*[*of 4* / *ε*] *ε* **by** *auto*
hence *N1* ≠ *0* **and** *N1*: *1* / *real N1* < *ε* / *4* **using** *ε*
by (*auto simp*: *divide-simps mult-ac intro*: *Nat.gr0I*)

have |*a n*| < *ε* **if** *n*: *n* ≥ *2* ∗ *N0* + *N1* + *7* **for** *n*
proof −
define *k* **where** *k* = ⌊*n* ∗ *ε*/*4*⌋
have *n* ∗ *ε* / *4* > *1* **and** *n* ∗ *ε* / *4* ≤ *n* / *4* **and** *n* / *4* < *n*
using *less-le-trans*[*OF N1*, *of n* / *N1* ∗ *ε* / *4*] ‹*N1* ≠ *0*› *ε n* **by** (*auto simp*: *field-simps*)
hence *k*: *k* > *0 4* ∗ *k* ≤ *n nat k* < *n* (*n* ∗ *ε* / *4*) − *1* < *k k* ≤ (*n* ∗ *ε* / *4*)
unfolding *k-def* **by** *linarith*+

have −*a n* < *ε*
proof −
have *N0* ≤ *n* − *nat k*
using *n k* **by** *linarith*
then have ∗: |∑ *k* = *n* − *nat k* .. *n*. *a k* / *k*| < (*ε* / *8*)^2
using *N0* [*of n* − *nat k n*] **by** *simp*
have −*a n* ≤ (*ε* / *8*)^2 ∗ *n* / ⌊*n* ∗ *ε* / *4*⌋ + ⌊*n* ∗ *ε* / *4*⌋ / (*n* − *k*)
using *sum-goestozero-lemma* [*OF* ∗ *le*, *THEN conjunct2*] *k* **by** (*simp add*: *of-nat-diff k-def*)
also have ... < *ε*
proof −
have *ε* / *16* ∗ *n* / *k* < *2*
using *k* **by** (*auto simp*: *field-simps*)
then have *ε* ∗ (*ε* / *16* ∗ *n* / *k*) < *ε* ∗ *2*
using *ε mult-less-cancel-left-pos* **by** *blast*
then have (*ε* / *8*)^2 ∗ *n* / *k* < *ε* / *2*
by (*simp add*: *field-simps power2-eq-square*)
moreover have *k* / (*n* − *k*) < *ε* / *2*
proof −
have (*ε* + *2*) ∗ *k* < *4* ∗ *k* **using** *k ε* **by** *simp*
also have ... ≤ *ε* ∗ *real n* **using** *k* **by** (*auto simp*: *field-simps*)
finally show *?thesis* **using** *k* **by** (*auto simp*: *field-simps*)
qed
ultimately show *?thesis* **unfolding** *k-def* **by** *linarith*
qed
finally show *?thesis* **.**

```
    qed
    moreover have a n < ε
    proof −
      have N0 ≤ n using n k by linarith
      then have ∗: |∑ k = n .. n + nat k. a k / k| < (ε/8)^2
        using N0 [of n n + nat k] by simp
      have a n ≤ (ε/8)^2 ∗ (n + nat k) / k + k / n
       using sum-goestozero-lemma [OF ∗ le, THEN conjunct1] k by (simp add:
of-nat-diff)
      also have ... < ε
      proof −
        have 4 ≤ 28 ∗ real-of-int k using k by linarith
        then have ε/16 ∗ n / k < 2 using k by (auto simp: field-simps)
        have ε ∗ (real n + k) < 32 ∗ k
        proof −
          have ε ∗ n / 4 < k + 1 by (simp add: mult.commute k-def)
          then have ε ∗ n < 4 ∗ k + 4 by (simp add: divide-simps)
          also have ... ≤ 8 ∗ k using k by auto
          finally have 1: ε ∗ real n < 8 ∗ k .
          have 2: ε ∗ k < k using k ε by simp
            show ?thesis using k add-strict-mono [OF 1 2] by (simp add: alge-
bra-simps)
      qed
        then have (ε / 8)^2 ∗ real (n + nat k) / k < ε / 2
          using ε k by (simp add: divide-simps mult-less-0-iff power2-eq-square)
        moreover have k / n < ε / 2
          using k ε by (auto simp: k-def field-simps)
        ultimately show ?thesis by linarith
      qed
      finally show ?thesis .
    qed
    ultimately show ?thesis by force
  qed
  then show ?thesis by blast
 qed
 show ∃ n0. ∀ n≥n0. |a n| < r
  using ∗ [of min r (1/5)] ‹0 < r› by force
qed
```

This leads us to the main intermediate result:

```
lemma Mertens-convergent: convergent (λn::nat. 𝔐 n − ln n)
proof −
 obtain c where c: summable (λn. (𝔐 n − ln n + c) / n)
   by (blast intro: mertens-summable)
 then obtain l where l: (λn. (𝔐 n − ln n + c) / n) sums l
   by (auto simp: summable-def)
 have ∗: (λn. 𝔐 n − ln n + c) ⟶ 0
   by (rule sum-goestozero-theorem[OF c]) auto
 hence (λn. 𝔐 n − ln n) ⟶ −c
```

by (*simp add*: *tendsto-iff dist-norm*)
thus *?thesis* **by** (*rule convergentI*)
qed

corollary $\mathfrak{M}$*-minus-ln-limit*:
obtains *c* **where** ((λ*x*::*real*. $\mathfrak{M}$ *x* − *ln x*) ⟶ *c*) *at-top*
proof −
from *Mertens-convergent* **obtain** *c* **where** (λ*n*. $\mathfrak{M}$ *n* − *ln n*) ⟶ *c*
by (*auto simp*: *convergent-def*)
hence *1*: ((λ*x*::*real*. $\mathfrak{M}$ (*nat* ⌊*x*⌋) − *ln* (*nat* ⌊*x*⌋)) ⟶ *c*) *at-top*
by (*rule filterlim-compose*) *real-asymp*
have *2*: ((λ*x*::*real*. *ln* (*nat* ⌊*x*⌋) − *ln x*) ⟶ *0*) *at-top*
by *real-asymp*
have *3*: ((λ*x*. $\mathfrak{M}$ *x* − *ln x*) ⟶ *c*) *at-top*
using *tendsto-add*[*OF 1 2*] **by** *simp*
with *that* **show** *?thesis* **by** *blast*
qed

4.3 The asymptotics of the prime-counting functions

We will now use the above result to prove the asymptotics of the prime-counting functions $\vartheta(x) \sim x$, $\psi(x) \sim x$, and $\pi(x) \sim x/\ln x$. The last of these is typically called the Prime Number Theorem, but since these functions can be expressed in terms of one another quite easily, knowing the asymptotics of any of them immediately gives the asymptotics of the other ones.

In this sense, all of the above are equivalent formulations of the Prime Number Theorem. The one we shall tackle first, due to its strong connection to the $\mathfrak{M}$ function, is $\vartheta(x) \sim x$.

We know that $\mathfrak{M}(x)$ has the asymptotic expansion $\mathfrak{M}(x) = \ln x + c + o(1)$. We also know that

$$\vartheta(x) = x\mathfrak{M}(x) - \int_2^x \mathfrak{M}(t)\,\mathrm{d}t\ .$$

Substituting in the above asymptotic equation, we obtain:

$$\begin{aligned}\vartheta(x) &= x\ln x + cx + o(x) - \int_2^x \ln t + c + o(1)\,\mathrm{d}t\\ &= x\ln x + cx + o(x) - (x\ln x - x + cx + o(x))\\ &= x + o(x)\end{aligned}$$

In conclusion, $\vartheta(x) \sim x$.

theorem ϑ*-asymptotics*: $\vartheta \sim[\mathit{at\text{-}top}]\ (\lambda x.\ x)$
proof −
from $\mathfrak{M}$*-minus-ln-limit* **obtain** *c* **where** *c*: ((λ*x*. $\mathfrak{M}$ *x* − *ln x*) ⟶ *c*) *at-top*
by *auto*
define *r* **where** *r* = (λ*x*. $\mathfrak{M}$ *x* − *ln x* − *c*)

```
  have 𝔐-expand: 𝔐 = (λx. ln x + c + r x)
    by (simp add: r-def)
  have r: r ∈ o(λ-. 1) unfolding r-def
    using tendsto-add[OF c tendsto-const[of −c]] by (intro smalloI-tendsto) auto

  define r' where r' = (λx. integral {2..x} r)
  have integrable-r: r integrable-on {x..y}
    if 2 ≤ x for x y :: real using that unfolding r-def
    by (intro integrable-diff integrable-primes-M)
       (auto intro!: integrable-continuous-real continuous-intros)
  hence integral: (r has-integral r' x) {2..x} if x ≥ 2 for x
    by (auto simp: has-integral-iff r'-def)
  have r': r' ∈ o(λx. x) using integrable-r unfolding r'-def
    by (intro integral-smallo[OF r]) (auto simp: filterlim-ident)

  define C where C = 2 * (c + ln 2 − 1)
  have ϑ ∼[at-top] (λx. x + (r x * x + C − r' x))
  proof (intro asymp-equiv-refl-ev eventually-mono[OF eventually-gt-at-top])
    fix x :: real assume x: x > 2
    have (𝔐 has-integral ((x * ln x − x + c * x) − (2 * ln 2 − 2 + c * 2) + r'
x)) {2..x}
      unfolding 𝔐-expand using x
      by (intro has-integral-add[OF fundamental-theorem-of-calculus integral])
         (auto simp flip: has-real-derivative-iff-has-vector-derivative
               intro!: derivative-eq-intros continuous-intros)
    from has-integral-unique[OF ϑ-conv-𝔐-integral this]
      show ϑ x = x + (r x * x + C − r' x) using x
      by (simp add: field-simps 𝔐-expand C-def)
  qed
  also have (λx. r x * x + C − r' x) ∈ o(λx. x)
  proof (intro sum-in-smallo r)
    show (λ-. C) ∈ o(λx. x) by real-asymp
  qed (insert landau-o.small-big-mult[OF r, of λx. x] r', simp-all)
  hence (λx. x + (r x * x + C − r' x)) ∼[at-top] (λx. x)
    by (subst asymp-equiv-add-right) auto
  finally show ?thesis by auto
qed
```

The various other forms of the Prime Number Theorem follow as simple corollaries.

```
corollary ψ-asymptotics: ψ ∼[at-top] (λx. x)
  using ϑ-asymptotics PNT4-imp-PNT5 by simp

corollary prime-number-theorem: π ∼[at-top] (λx. x / ln x)
  using ϑ-asymptotics PNT4-imp-PNT1 by simp

corollary ln-π-asymptotics: (λx. ln (π x)) ∼[at-top] ln
  using prime-number-theorem PNT1-imp-PNT1' by simp
```

corollary *π-ln-π-asymptotics*: $(\lambda x.\ \pi\ x * ln\ (\pi\ x)) \sim[at\text{-}top]\ (\lambda x.\ x)$
using *prime-number-theorem PNT1-imp-PNT2* **by** *simp*

corollary *nth-prime-asymptotics*: $(\lambda n.\ real\ (nth\text{-}prime\ n)) \sim[at\text{-}top]\ (\lambda n.\ real\ n * ln\ (real\ n))$
using *π-ln-π-asymptotics PNT2-imp-PNT3* **by** *simp*

The following versions use a little less notation.

corollary *prime-number-theorem′*: $((\lambda x.\ \pi\ x\ /\ (x\ /\ ln\ x)) \longrightarrow 1)\ at\text{-}top$
using *prime-number-theorem*
by (*rule asymp-equivD-strong*[*OF - eventually-mono*[*OF eventually-gt-at-top*[*of 1*]]]) *auto*

corollary *prime-number-theorem″*:
$(\lambda x.\ card\ \{p.\ prime\ p \wedge real\ p \leq x\}) \sim[at\text{-}top]\ (\lambda x.\ x\ /\ ln\ x)$
proof −
have $\pi = (\lambda x.\ card\ \{p.\ prime\ p \wedge real\ p \leq x\})$
by (*intro ext*) (*simp add*: *π-def prime-sum-upto-def*)
with *prime-number-theorem* **show** *?thesis* **by** *simp*
qed

corollary *prime-number-theorem‴*:
$(\lambda n.\ card\ \{p.\ prime\ p \wedge p \leq n\}) \sim[at\text{-}top]\ (\lambda n.\ real\ n\ /\ ln\ (real\ n))$
proof −
have $(\lambda n.\ card\ \{p.\ prime\ p \wedge real\ p \leq real\ n\}) \sim[at\text{-}top]\ (\lambda n.\ real\ n\ /\ ln\ (real\ n))$
using *prime-number-theorem″*
by (*rule asymp-equiv-compose′*) (*simp add*: *filterlim-real-sequentially*)
thus *?thesis* **by** *simp*
qed

end

5 Mertens' Theorems

theory *Mertens-Theorems*
imports
Prime-Counting-Functions
Stirling-Formula.Stirling-Formula
begin

In this section, we will prove Mertens' First and Second Theorem. These are weaker results than the Prime Number Theorem, and we will derive them without using it.

However, like Mertens himself, we will not only prove them *asymptotically*, but *absolutely*. This means that we will show that the remainder terms are not only "Big-O" of some bound, but we will give concrete (and reasonably

tight) upper and lower bounds for them that hold on the entire domain. This makes the proofs a bit more tedious.

5.1 Absolute Bounds for Mertens' First Theorem

We have already shown the asymptotic form of Mertens' first theorem, i. e. $\mathfrak{M}(n) = \ln n + O(1)$. We now want to obtain some absolute bounds on the $O(1)$ remainder term using a more careful derivation than before.

The precise bounds we will show are $\mathfrak{M}(n) - \ln n \in (-1 - \frac{9}{\pi^2}; \ln 4] \approx (-1.9119; 1.3863]$ for $n \in \mathbb{N}$.

First, we need a simple lemma on the finiteness of exponents to consider in a sum of all prime powers up to a certain point:

```
lemma exponents-le-finite:
  assumes p > (1 :: nat) k > 0
  shows   finite {i. real (p ^ (k * i + l)) ≤ x}
proof (rule finite-subset)
  show {i. real (p ^ (k * i + l)) ≤ x} ⊆ {..nat ⌊x⌋}
  proof safe
    fix i assume i: real (p ^ (k * i + l)) ≤ x
    have i < 2 ^ i by (rule less-exp)
    also from assms have i ≤ k * i + l by (cases k) auto
    hence 2 ^ i ≤ (2 ^ (k * i + l) :: nat)
      using assms by (intro power-increasing) auto
    also have ... ≤ p ^ (k * i + l) using assms by (intro power-mono) auto
    also have real ... ≤ x using i by simp
    finally show i ≤ nat ⌊x⌋ by linarith
  qed
qed auto
```

Next, we need the following bound on $\zeta'(2)$:

```
lemma deriv-zeta-2-bound: Re (deriv zeta 2) > -1
proof -
  have ((λx::real. ln (x + 3) * (x + 3) powr -2) has-integral (ln 3 + 1) / 3)
(interior {0..})
    using ln-powr-has-integral-at-top[of 1 0 3 -2]
    by (simp add: interior-real-atLeast powr-minus)
  hence ((λx::real. ln (x + 3) * (x + 3) powr -2) has-integral (ln 3 + 1) / 3)
{0..}
    by (subst (asm) has-integral-interior) auto
  also have ?this ⟷ ((λx::real. ln (x + 3) / (x + 3) ^ 2) has-integral (ln 3 +
1) / 3) {0..}
    by (intro has-integral-cong) (auto simp: powr-minus field-simps)
  finally have int: ... .
  have exp (1 / 2 :: real) ^ 2 ≤ 2 ^ 2
    using exp-le by (subst exp-double [symmetric]) simp-all
  hence exp-half: exp (1 / 2 :: real) ≤ 2
    by (rule power2-le-imp-le) auto
```

```
  have mono: ln x / x ^ 2 ≤ ln y / y ^ 2 if y ≥ exp (1/2) x ≥ y for x y :: real
  proof (rule DERIV-nonpos-imp-nonincreasing[of - - λx. ln x / x ^ 2])
    fix t assume t: t ≥ y t ≤ x
    have y > 0 by (rule less-le-trans[OF - that(1)]) auto
    with t that have ln t ≥ ln (exp (1 / 2))
      by (subst ln-le-cancel-iff) auto
    hence ln t ≥ 1 / 2 by (simp only: ln-exp)
    from t ‹y > 0› have ((λx. ln x / x ^ 2) has-field-derivative ((1 - 2 * ln t) /
t ^ 3)) (at t)
      by (auto intro!: derivative-eq-intros simp: eval-nat-numeral field-simps)
    moreover have (1 - 2 * ln t) / t ^ 3 ≤ 0
      using t that ‹y > 0› ‹ln t ≥ 1 / 2› by (intro divide-nonpos-pos) auto
    ultimately show ∃f'. ((λx. ln x / x ^ 2) has-field-derivative f') (at t) ∧ f' ≤
0 by blast
  qed fact+

  have fds-converges (fds-deriv fds-zeta) (2 :: complex)
    by (intro fds-converges-deriv) auto
  hence (λn. of-real (-ln (real (Suc n)) / (of-nat (Suc n)) ^ 2)) sums deriv zeta
2
      by (auto simp: fds-converges-altdef add-ac eval-fds-deriv-zeta fds-nth-deriv
scaleR-conv-of-real
          simp del: of-nat-Suc)
  note * = sums-split-initial-segment[OF sums-minus[OF sums-Re[OF this]], of 3]
    have (λn. ln (real (n+4)) / real (n+4) ^ 2) sums (-Re (deriv zeta 2) - (ln
2 / 4 + ln 3 / 9))
      using * by (simp add: eval-nat-numeral)
  hence -Re (deriv zeta 2) - (ln 2 / 4 + ln 3 / 9) =
          (∑ n. ln (real (Suc n) + 3) / (real (Suc n) + 3) ^ 2)
    by (simp-all add: sums-iff algebra-simps)
  also have ... ≤ (ln 3 + 1) / 3 using int exp-half
   by (intro decreasing-sum-le-integral divide-nonneg-pos mono) (auto simp: powr-minus
field-simps)
  finally have -Re (deriv zeta 2) ≤ (16 * ln 3 + 9 * ln 2 + 12) / 36
    by simp
  also have ln 3 ≤ (11 / 10 :: real)
   using ln-approx-bounds[of 3 2] by (simp add: power-numeral-reduce numeral-2-eq-2)
  hence (16 * ln 3 + 9 * ln 2 + 12) / 36 ≤ (16 * (11 / 10) + 9 * 25 / 36 +
12) / (36 :: real)
    using ln2-le-25-over-36 by (intro add-mono mult-left-mono divide-right-mono)
auto
  also have ... < 1 by simp
  finally show ?thesis by simp
qed
```

Using the logarithmic derivative of Euler's product formula for $\zeta(s)$ at $s = 2$

and the bound on $\zeta'(2)$ we have just derived, we can obtain the bound

$$\sum_{p^i \leq x, i \geq 2} \frac{\ln p}{p^i} < \frac{9}{\pi^2} \ .$$

lemma *mertens-remainder-aux-bound*:
  fixes x :: *real*
  defines $R \equiv (\sum (p,i) \mid$ *prime* $p \wedge i > 1 \wedge$ *real* $(p \hat{} i) \leq x$. *ln* (*real* p) / $p \hat{} i)$
  shows $R < 9$ / pi^2
proof −
  define S' **where** $S' = \{(p, i).$ *prime* $p \wedge i > 1 \wedge$ *real* $(p \hat{} i) \leq x\}$
  define S'' **where** $S'' = \{(p, i).$ *prime* $p \wedge i > 1 \wedge$ *real* $(p \hat{} Suc\ i) \leq x\}$

  have *finite-row*: *finite* $\{i.\ i > 1 \wedge$ *real* $(p \hat{} (i + k)) \leq x\}$ **if** *p*: *prime* p **for** $p\ k$
  proof (*rule finite-subset*)
    show $\{i.\ i > 1 \wedge$ *real* $(p \hat{} (i + k)) \leq x\} \subseteq \{..nat\ \lfloor x \rfloor\}$
    proof *safe*
      fix i **assume** *i*: $i > 1$ *real* $(p \hat{} (i + k)) \leq x$
      have $i < 2 \hat{} (i + k)$ **by** (*induction i*) *auto*
        also from p **have** $\ldots \leq p \hat{} (i + k)$ **by** (*intro power-mono*) (*auto dest*: *prime-gt-1-nat*)
      also have *real* $\ldots \leq x$ **using** i **by** *simp*
      finally show $i \leq nat\ \lfloor x \rfloor$ **by** *linarith*
    qed
  qed *auto*

  have $S'' \subseteq S'$ **unfolding** S''-*def* S'-*def*
  proof *safe*
    fix $p\ i$ **assume** *pi*: *prime* p *real* $(p \hat{} Suc\ i) \leq x\ i > 1$
    have *real* $(p \hat{} i) \leq$ *real* $(p \hat{} Suc\ i)$
      using *pi* **unfolding** *of-nat-le-iff* **by** (*intro power-increasing*) (*auto dest*: *prime-gt-1-nat*)
    also have $\ldots \leq x$ **by** *fact*
    finally show *real* $(p \hat{} i) \leq x$ **.**
  qed

  have S'-*alt*: $S' = (SIGMA\ p{:}\{p.$ *prime* $p \wedge$ *real* $p \leq x\}.\ \{i.\ i > 1 \wedge$ *real* $(p \hat{} i) \leq x\})$
    unfolding S'-*def*
  proof *safe*
    fix $p\ i$ **assume** *prime* p *real* $(p \hat{} i) \leq x\ i > 1$
    hence $p \hat{} 1 \leq p \hat{} i$
      by (*intro power-increasing*) (*auto dest*: *prime-gt-1-nat*)
    also have *real* $\ldots \leq x$ **by** *fact*
    finally show *real* $p \leq x$ **by** *simp*
  qed

  have *finite*: *finite* $\{p.$ *prime* $p \wedge$ *real* $p \leq x\}$
    by (*rule finite-subset*[*OF - finite-Nats-le-real*[*of x*]]) (*auto dest*: *prime-gt-0-nat*)

```
  have finite S' unfolding S'-alt using finite-row[of - 0]
    by (intro finite-SigmaI finite) auto

  have R ≤ 3 / 2 * (∑(p, i) | (p, i) ∈ S' ∧ even i. ln (real p) / real (p ^ i))
  proof −
    have R = (∑y∈{0, 1}. ∑z | z ∈ S' ∧ snd z mod 2 = y. ln (real (fst z)) /
real (fst z ^ snd z))
      using ‹finite S'› by (subst sum.group) (auto simp: case-prod-unfold R-def
S'-def)
    also have ... = (∑(p,i) | (p, i) ∈ S' ∧ even i. ln (real p) / real (p ^ i)) +
                    (∑(p,i) | (p, i) ∈ S' ∧ odd i. ln (real p) / real (p ^ i))
      unfolding even-iff-mod-2-eq-zero odd-iff-mod-2-eq-one by (simp add: case-prod-unfold)
    also have (∑(p,i) | (p, i) ∈ S' ∧ odd i. ln (real p) / real (p ^ i)) =
                (∑(p,i) | (p, i) ∈ S'' ∧ even i. ln (real p) / real (p ^ Suc i))
      by (intro sum.reindex-bij-witness[of - λ(p,i). (p, Suc i) λ(p,i). (p, i − 1)])
        (auto simp: case-prod-unfold S'-def S''-def elim: oddE simp del: power-Suc)
    also have ... ≤ (∑(p,i) | (p, i) ∈ S' ∧ even i. ln (real p) / real (p ^ Suc i))
      using ‹S'' ⊆ S'› unfolding case-prod-unfold
      by (intro sum-mono2 divide-nonneg-pos ln-ge-zero finite-subset[OF - ‹finite
S'›])
        (auto simp: S'-def S''-def case-prod-unfold dest: prime-gt-0-nat simp del:
power-Suc)
    also have ... ≤ (∑(p,i) | (p, i) ∈ S' ∧ even i. ln (real p) / real (2 * p ^ i))
      unfolding case-prod-unfold
      by (intro sum-mono divide-left-mono) (auto simp: S'-def dest!: prime-gt-1-nat)
    also have ... = (1 / 2) * (∑(p,i) | (p, i) ∈ S' ∧ even i. ln (real p) / real (p
^ i))
      by (subst sum-distrib-left) (auto simp: case-prod-unfold)
    also have (∑(p,i) | (p, i) ∈ S' ∧ even i. ln (real p) / real (p ^ i)) + ... =
                3 / 2 * (∑(p,i) | (p, i) ∈ S' ∧ even i. ln (real p) / real (p ^ i))
      by simp
    finally show ?thesis by simp
  qed

  also have (∑(p,i) | (p, i) ∈ S' ∧ even i. ln (real p) / real (p ^ i)) =
              (∑p | prime p ∧ real p ≤ x. ln (real p) *
                (∑i | i > 0 ∧ even i ∧ real (p ^ i) ≤ x. (1 / real p) ^ i))
   unfolding sum-distrib-left
  proof (subst sum.Sigma[OF - ballI])
    fix p assume p: p ∈ {p. prime p ∧ real p ≤ x}
    thus finite {i. 0 < i ∧ even i ∧ real (p ^ i) ≤ x}
      by (intro finite-subset[OF - exponents-le-finite[of p 1 0 x]]) (auto dest:
prime-gt-1-nat)
  qed (auto intro!: sum.cong finite-subset[OF - finite-Nats-le-real[of x]]
        dest: prime-gt-0-nat simp: S'-alt power-divide)
  also have ... ≤ (∑p | prime p ∧ real p ≤ x. ln (real p) / (real p ^ 2 − 1))
  proof (rule sum-mono)
    fix p assume p: p ∈ {p. prime p ∧ real p ≤ x}
    have p > 1 using p by (auto dest: prime-gt-1-nat)
```

```
  have (∑ i | i > 0 ∧ even i ∧ real (p ^ i) ≤ x. (1 / real p) ^ i) =
        (∑ i | real (p ^ (2 * i + 2)) ≤ x. (1 / real p) ^ (2 * i)) / real p ^ 2
    (is - = ?S / -) unfolding sum-divide-distrib
    by (rule sum.reindex-bij-witness[of - λi. 2 * Suc i λi. (i - 2) div 2])
      (insert ‹p > 1›, auto simp: numeral-3-eq-3 power2-eq-square power-diff
          algebra-simps elim!: evenE)
  also have ?S = (∑ i | real (p ^ (2 * i + 2)) ≤ x. (1 / real p ^ 2) ^ i)
    by (subst power-mult) (simp-all add: algebra-simps power-divide)
  also have ... ≤ (∑ i. (1 / real p ^ 2) ^ i)
    using exponents-le-finite[of p 2 2 x] ‹p > 1›
    by (intro sum-le-suminf) (auto simp: summable-geometric-iff)
  also have ... = real p ^ 2 / (real p ^ 2 - 1)
    using ‹p > 1› by (subst suminf-geometric) (auto simp: field-simps)
  also have ... / real p ^ 2 = 1 / (real p ^ 2 - 1)
    using ‹p > 1 › by (simp add: divide-simps)
  finally have (∑ i | 0 < i ∧ even i ∧ real (p ^ i) ≤ x. (1 / real p) ^ i) ≤
                1 / (real p ^ 2 - 1) (is ?lhs ≤ ?rhs)
    using ‹p > 1› by (simp add: divide-right-mono)
  thus ln (real p) * ?lhs ≤ ln (real p) / (real p ^ 2 - 1)
    using ‹p > 1› by (simp add: divide-simps)
qed
also have ... = (∑ a p | prime p ∧ real p ≤ x. ln (real p) / (real p ^ 2 - 1))
  using finite by (intro infsetsum-finite [symmetric]) auto
also have ... ≤ (∑ a p | prime p. ln (real p) / (real p ^ 2 - 1))
  using eval-fds-logderiv-zeta-real[of 2] finite
    by (intro infsetsum-mono-neutral-left divide-nonneg-pos) (auto simp: dest:
prime-gt-1-nat)
also have ... = -Re (deriv zeta (of-real 2) / zeta (of-real 2))
  by (subst eval-fds-logderiv-zeta-real) auto
also have ... = (-Re (deriv zeta 2)) * (6 / pi^2)
  by (simp add: zeta-even-numeral)
also have ... < 1 * (6 / pi^2)
  using deriv-zeta-2-bound by (intro mult-strict-right-mono) auto
also have 3 / 2 * ... = 9 / pi^2 by simp
finally show ?thesis by simp
qed
```

We now consider the equation

$$\ln(n!) = \sum_{k \leq n} \Lambda(k) \left\lfloor \frac{n}{k} \right\rfloor$$

and estimate both sides in different ways. The left-hand-side can be estimated using Stirling's formula, and we can simplify the right-hand side to

$$\sum_{k \leq n} \Lambda(k) \left\lfloor \frac{n}{k} \right\rfloor = \sum_{p^i \leq x, i \geq 1} \ln p \left\lfloor \frac{n}{p^i} \right\rfloor$$

and then split the sum into those p^i with $i = 1$ and those with $i \geq 2$. Applying the bound we have just shown and some more routine estimates,

we obtain the following reasonably strong version of Mertens' First Theorem on the naturals: $\mathfrak{M}(n) - \ln(n) \in (-1 - \frac{9}{\pi^2}; \ln 4]$

```
theorem mertens-bound-strong:
  fixes n :: nat assumes n: n > 0
  shows   𝔐 n − ln n ∈ {−1 − 9 / pi²<..ln 4}
proof (cases n ≥ 3)
  case False
  with n consider n = 1 | n = 2 by force
  thus ?thesis
  proof cases
    assume [simp]: n = 1
    have −1 + (−9 / pi²) < 0
      by (intro add-neg-neg divide-neg-pos) auto
    thus ?thesis by simp
  next
    assume [simp]: n = 2
    have eq: 𝔐 n − ln n = −ln 2 / 2 by (simp add: eval-𝔐)
    have −1 − 9 / pi ^ 2 + ln 2 / 2 ≤ −1 − 9 / 4 ^ 2 + 25 / 36 / 2
      using pi-less-4 ln2-le-25-over-36
     by (intro diff-mono add-mono divide-left-mono divide-right-mono power-mono)
auto
    also have ... < 0 by simp
    finally have −ln 2 / 2 > −1 − 9 / pi² by simp
    moreover {
      have −ln 2 / 2 ≤ (0::real) by (intro divide-nonpos-pos) auto
      also have ... ≤ ln 4 by simp
      finally have −ln 2 / 2 ≤ ln (4 :: real) by simp
    }
    ultimately show ?thesis unfolding eq by simp
  qed

next
  case True
  hence n: n ≥ 3 by simp
  have finite: finite {(p, i). prime p ∧ i ≥ 1 ∧ p ^ i ≤ n}
  proof (rule finite-subset)
    show {(p, i). prime p ∧ i ≥ 1 ∧ p ^ i ≤ n}
          ⊆ {..nat ⌊root 1 (real n)⌋} × {..nat ⌊log 2 (real n)⌋}
      using primepows-le-subset[of real n 1] n unfolding of-nat-le-iff by auto
  qed auto

  define r where r = prime-sum-upto (λp. ln (real p) * frac (real n / real p)) n
  define R where R = (∑(p,i) | prime p ∧ i > 1 ∧ p ^ i ≤ n. ln (real p) * real
(n div (p ^ i)))
  define R' where R' = (∑(p,i) | prime p ∧ i > 1 ∧ p ^ i ≤ n. ln (real p) / p
^ i)
  have [simp]: ln (4 :: real) = 2 * ln 2
    using ln-realpow[of 2 2] by simp
  from pi-less-4 have ln pi ≤ ln 4 by (subst ln-le-cancel-iff) auto
```

```
  also have ... = 2 * ln 2 by simp
  also have ... ≤ 2 * (25 / 36) by (intro mult-left-mono ln2-le-25-over-36) auto
  finally have ln-pi: ln pi ≤ 25 / 18 by simp
  have ln 3 ≤ ln (4::nat) by (subst ln-le-cancel-iff) auto
  also have ... = 2 * ln 2 by simp
  also have ... ≤ 2 * (25 / 36) by (intro mult-left-mono ln2-le-25-over-36) auto
  finally have ln-3: ln (3::real) ≤ 25 / 18 by simp

  have R / n = (∑(p,i) | prime p ∧ i > 1 ∧ p ^ i ≤ n. ln (real p) * (real (n div
(p ^ i)) / n))
    by (simp add: R-def sum-divide-distrib field-simps case-prod-unfold)
  also have ... ≤ (∑(p,i) | prime p ∧ i > 1 ∧ p ^ i ≤ n. ln (real p) * (1 / p ^
i))
    unfolding R'-def case-prod-unfold using n
   by (intro sum-mono mult-left-mono) (auto simp: field-simps real-of-nat-div dest:
prime-gt-0-nat)
  also have ... = R' by (simp add: R'-def)
  also have R' < 9 / pi^2
    unfolding R'-def using mertens-remainder-aux-bound[of n] by simp
  finally have R / n < 9 / pi^2 .
  moreover have R ≥ 0
   unfolding R-def by (intro sum-nonneg mult-nonneg-nonneg) (auto dest: prime-gt-0-nat)
  ultimately have R-bounds: R / n ∈ {0..<9 / pi^2} by simp

  have ln (fact n :: real) ≤ ln (2 * pi * n) / 2 + n * ln n - n + 1 / (12 * n)
    using ln-fact-bounds(2)[of n] n by simp
  also have ... / n - ln n = -1 + (ln 2 + ln pi) / (2 * n) + (ln n / n) / 2 +
1 / (12 * real n ^ 2)
    using n by (simp add: power2-eq-square field-simps ln-mult)
  also have ... ≤ -1 + (ln 2 + ln pi) / (2 * 3) + (ln 3 / 3) / 2 + 1 / (12 *
3^2)
    using exp-le n pi-gt3
    by (intro add-mono divide-right-mono divide-left-mono mult-mono
          mult-pos-pos ln-x-over-x-mono power-mono) auto
  also have ... ≤ -1 + (25 / 36 + 25 / 18) / (2 * 3) + (25 / 18 / 3) / 2 +
1 / (12 * 3^2)
     using ln-pi ln2-le-25-over-36 ln-3 by (intro add-mono divide-left-mono di-
vide-right-mono) auto
  also have ... ≤ 0 by simp
 finally have ln n - ln (fact n) / n ≥ 0 using n by (simp add: divide-right-mono)
  have -ln (fact n) ≤ -ln (2 * pi * n) / 2 - n * ln n + n
    using ln-fact-bounds(1)[of n] n by simp
  also have ln n + ... / n = -ln (2 * pi) / (2 * n) - (ln n / n) / 2 + 1
    using n by (simp add: field-simps ln-mult)
  also have ... ≤ 0 - 0 + 1
    using pi-gt3 n by (intro add-mono diff-mono) auto
  finally have upper: ln n - ln (fact n) / n ≤ 1
    using n by (simp add: divide-right-mono)
  with ‹ln n - ln (fact n) / n ≥ 0› have fact-bounds: ln n - ln (fact n) / n ∈
```

$\{0..1\}$ **by** *simp*

have $r \leq$ *prime-sum-upto* $(\lambda p.\ ln\ p * 1)\ n$
using *less-imp-le*[*OF frac-lt-1*] **unfolding** *r-def* ϑ*-def prime-sum-upto-def*
by (*intro sum-mono mult-left-mono*) (*auto simp*: *dest*: *prime-gt-0-nat*)
also have $\ldots = \vartheta\ n$ **by** (*simp add*: ϑ*-def*)
also have $\ldots < ln\ 4 * n$ **using** *n* **by** (*intro* ϑ*-upper-bound*) *auto*
finally have $r\ /\ n < ln\ 4$ **using** *n* **by** (*simp add*: *field-simps*)
moreover have $r \geq 0$ **unfolding** *r-def prime-sum-upto-def*
by (*intro sum-nonneg mult-nonneg-nonneg*) (*auto dest*: *prime-gt-0-nat*)
ultimately have *r-bounds*: $r\ /\ n \in \{0..<ln\ 4\}$ **by** *simp*

have $ln\ (fact\ n :: real) = sum\text{-}upto\ (\lambda k.\ mangoldt\ k * real\ (n\ div\ k))\ (real\ n)$
by (*simp add*: *ln-fact-conv-sum-upto-mangoldt*)
also have $\ldots = (\sum (p,i) \mid prime\ p \wedge i > 0 \wedge real\ (p\ \hat{}\ i) \leq real\ n.$
$ln\ (real\ p) * real\ (n\ div\ (p\ \hat{}\ i)))$
by (*intro sum-upto-primepows*) (*auto simp*: *mangoldt-non-primepow*)
also have $\{(p,\ i).\ prime\ p \wedge i > 0 \wedge real\ (p\ \hat{}\ i) \leq real\ n\} =$
$\{(p,\ i).\ prime\ p \wedge i = 1 \wedge p \leq n\} \cup$
$\{(p,\ i).\ prime\ p \wedge i > 1 \wedge (p\ \hat{}\ i) \leq n\}$ **unfolding** *of-nat-le-iff*
by (*auto simp*: *not-less le-Suc-eq*)
also have $(\sum (p,i) \in \ldots.\ ln\ (real\ p) * real\ (n\ div\ (p\ \hat{}\ i))) =$
$(\sum (p,i) \mid prime\ p \wedge i = 1 \wedge p \leq n.\ ln\ (real\ p) * real\ (n\ div\ (p\ \hat{}\ i)))$
$+\ R$
(**is** - = *?S* + -)
by (*subst sum.union-disjoint*) (*auto intro*!: *finite-subset*[*OF - finite*] *simp*: *R-def*)
also have *?S* = *prime-sum-upto* $(\lambda p.\ ln\ (real\ p) * real\ (n\ div\ p))\ n$
unfolding *prime-sum-upto-def*
by (*intro sum.reindex-bij-witness*[*of - λp. (p, 1) fst*]) *auto*
also have $\ldots =$ *prime-sum-upto* $(\lambda p.\ ln\ (real\ p) * real\ n\ /\ real\ p)\ n - r$
unfolding *r-def prime-sum-upto-def sum-subtractf*[*symmetric*] **using** *n*
by (*intro sum.cong*) (*auto simp*: *frac-def real-of-nat-div algebra-simps*)
also have *prime-sum-upto* $(\lambda p.\ ln\ (real\ p) * real\ n\ /\ real\ p)\ n = n * \mathfrak{M}\ n$
by (*simp add*: *primes-M-def sum-distrib-left sum-distrib-right prime-sum-upto-def field-simps*)
finally have $\mathfrak{M}\ n - ln\ n = ln\ (fact\ n)\ /\ n - ln\ n + r\ /\ n - R\ /\ n$
using *n* **by** (*simp add*: *field-simps*)
hence $ln\ n - \mathfrak{M}\ n = ln\ n - ln\ (fact\ n)\ /\ n - r\ /\ n + R\ /\ n$
by *simp*
with *fact-bounds r-bounds R-bounds* **show** $\mathfrak{M}\ n - ln\ n \in \{-1 - 9\ /\ pi^2 <..ln\ 4\}$
by *simp*
qed

As a simple corollary, we obtain a similar bound on the reals.

lemma *mertens-bound-real-strong*:
fixes *x* :: *real* **assumes** *x*: $x \geq 1$
shows $\mathfrak{M}\ x - ln\ x \in \{-1 - 9\ /\ pi\ \hat{}\ 2 - ln\ (1 + frac\ x\ /\ real\ (nat\ \lfloor x \rfloor)) <..$
$ln\ 4\}$

proof −
  have $\mathfrak{M}\ x - ln\ x \leq \mathfrak{M}\ (real\ (nat\ \lfloor x \rfloor)) - ln\ (real\ (nat\ \lfloor x \rfloor))$
    using *assms* **by** *simp*
  also have $\ldots \leq ln\ 4$
    using *mertens-bound-strong*[*of nat* $\lfloor x \rfloor$] *assms* **by** *simp*
  finally have $\mathfrak{M}\ x - ln\ x \leq ln\ 4$ **.**

  from *assms* **have** *pos*: *real-of-int* $\lfloor x \rfloor \neq 0$ **by** *linarith*
  have *frac x / real* $(nat\ \lfloor x \rfloor) \geq 0$
    using *assms* **by** (*intro divide-nonneg-pos*) *auto*
  moreover have *frac x / real* $(nat\ \lfloor x \rfloor) \leq 1\ /\ 1$
    using *assms frac-lt-1*[*of x*] **by** (*intro frac-le*) *auto*
  ultimately have $*$: *frac x / real* $(nat\ \lfloor x \rfloor) \in \{0..1\}$ **by** *auto*
  have $ln\ x - ln\ (real\ (nat\ \lfloor x \rfloor)) = ln\ (x\ /\ real\ (nat\ \lfloor x \rfloor))$
    using *assms ln-div pos* **by** *force*
  also have *x / real* $(nat\ \lfloor x \rfloor) = 1 + frac\ x\ /\ real\ (nat\ \lfloor x \rfloor)$
    using *assms pos* **by** (*simp add*: *frac-def field-simps*)
  finally have $\mathfrak{M}\ x - ln\ x > -1 - 9/pi\hat{}2 - ln\ (1 + frac\ x\ /\ real\ (nat\ \lfloor x \rfloor))$
    using *mertens-bound-strong*[*of nat* $\lfloor x \rfloor$] *x* **by** *simp*
  with ‹$\mathfrak{M}\ x - ln\ x \leq ln\ 4$› **show** *?thesis* **by** *simp*
qed

We weaken this estimate a bit to obtain nicer bounds:

lemma *mertens-bound-real′*:
  fixes *x* :: *real* **assumes** *x*: $x \geq 1$
  shows $\mathfrak{M}\ x - ln\ x \in \{-2<..25/18\}$
proof −
  have $\mathfrak{M}\ x - ln\ x \leq ln\ 4$
    using *mertens-bound-real-strong*[*of x*] *x* **by** *simp*
  also have $\ldots \leq 25\ /\ 18$
    using *ln-realpow*[*of 2 2*] *ln2-le-25-over-36* **by** *simp*
  finally have $\mathfrak{M}\ x - ln\ x \leq 25\ /\ 18$ **.**

  have *ln2*: $ln\ (2 :: real) \in \{2/3..25/36\}$
    using *ln-approx-bounds*[*of 2 1*] **by** (*simp add*: *eval-nat-numeral*)
  have *ln3*: $ln\ (3::real) \in \{1..10/9\}$
    using *ln-approx-bounds*[*of 3 1*] **by** (*simp add*: *eval-nat-numeral*)
  have *ln5*: $ln\ (5::real) \in \{4/3..76/45\}$
    using *ln-approx-bounds*[*of 5 1*] **by** (*simp add*: *eval-nat-numeral*)
  have *ln7*: $ln\ (7::real) \in \{3/2..15/7\}$
    using *ln-approx-bounds*[*of 7 1*] **by** (*simp add*: *eval-nat-numeral*)
  have *ln11*: $ln\ (11::real) \in \{5/3..290/99\}$
    using *ln-approx-bounds*[*of 11 1*] **by** (*simp add*: *eval-nat-numeral*)

— Choosing the lower bound -2 is somewhat arbitrary here; it is a trade-off between getting a reasonably tight bound and having to make lots of case distinctions. To get -2 as a lower bound, we have to show the cases up to $x = 11$ by case distinction,
  have $\mathfrak{M}\ x - ln\ x > -2$

```
  proof (cases x ≥ 11)
    case False
     hence x ∈ {1..<2} ∨ x ∈ {2..<3} ∨ x ∈ {3..<5} ∨ x ∈ {5..<7} ∨ x ∈
{7..<11}
      using x by force
    thus ?thesis
    proof (elim disjE)
      assume x: x ∈ {1..<2}
      hence ln x − 𝔐 x ≤ ln 2 − 0
        by (intro diff-mono) auto
      also have ... < 2 using ln2-le-25-over-36 by simp
      finally show ?thesis by simp
    next
      assume x: x ∈ {2..<3}
      hence [simp]: ⌊x⌋ = 2 by (intro floor-unique) auto
      from x have ln x − 𝔐 x ≤ ln 3 − ln 2 / 2
        by (intro diff-mono) (auto simp: eval-𝔐)
     also have ... = ln (9 / 2) / 2 using ln-realpow[of 3 2] by (simp add: ln-div)
       also have ... < 2 using ln-approx-bounds[of 9 / 2 1] by (simp add:
eval-nat-numeral)
      finally show ?thesis by simp
    next
      assume x: x ∈ {3..<5}
      hence 𝔐 3 = 𝔐 x
        unfolding primes-M-def
        by (intro prime-sum-upto-eqI'[where a' = 3 and b' = 4])
           (auto simp: nat-le-iff le-numeral-iff nat-eq-iff floor-eq-iff)
      also have 𝔐 3 = ln 2 / 2 + ln 3 / 3
        by (simp add: eval-𝔐 eval-nat-numeral mark-out-code)
      finally have [simp]: 𝔐 x = ln 2 / 2 + ln 3 / 3 ..
      from x have ln x − 𝔐 x ≤ ln 5 − (ln 2 / 2 + ln 3 / 3)
        by (intro diff-mono) auto
      also have ... < 2 using ln2 ln3 ln5 by simp
      finally show ?thesis by simp
    next
      assume x: x ∈ {5..<7}
      hence 𝔐 5 = 𝔐 x
        unfolding primes-M-def
        by (intro prime-sum-upto-eqI'[where a' = 5 and b' = 6])
           (auto simp: nat-le-iff le-numeral-iff nat-eq-iff floor-eq-iff)
      also have 𝔐 5 = ln 2 / 2 + ln 3 / 3 + ln 5 / 5
        by (simp add: eval-𝔐 eval-nat-numeral mark-out-code)
      finally have [simp]: 𝔐 x = ln 2 / 2 + ln 3 / 3 + ln 5 / 5 ..
      from x have ln x − 𝔐 x ≤ ln 7 − (ln 2 / 2 + ln 3 / 3 + ln 5 / 5)
        by (intro diff-mono) auto
      also have ... < 2 using ln2 ln3 ln5 ln7 by simp
      finally show ?thesis by simp
    next
      assume x: x ∈ {7..<11}
```

```
    hence 𝔐 7 = 𝔐 x
      unfolding primes-M-def
      by (intro prime-sum-upto-eqI'[where a' = 7 and b' = 10])
         (auto simp: nat-le-iff le-numeral-iff nat-eq-iff floor-eq-iff)
    also have 𝔐 7 = ln 2 / 2 + ln 3 / 3 + ln 5 / 5 + ln 7 / 7
      by (simp add: eval-𝔐 eval-nat-numeral mark-out-code)
    finally have [simp]: 𝔐 x = ln 2 / 2 + ln 3 / 3 + ln 5 / 5 + ln 7 / 7 ..
    from x have ln x - 𝔐 x ≤ ln 11 - (ln 2 / 2 + ln 3 / 3 + ln 5 / 5 + ln 7
/ 7)
      by (intro diff-mono) auto
    also have ... < 2 using ln2 ln3 ln5 ln7 ln11 by simp
    finally show ?thesis by simp
  qed
 next
  case True
  have ln x - 𝔐 x ≤ 1 + 9/pi^2 + ln (1 + frac x / real (nat ⌊x⌋))
    using mertens-bound-real-strong[of x] x by simp
  also have 1 + frac x / real (nat ⌊x⌋) ≤ 1 + 1 / 11
    using True frac-lt-1[of x] by (intro add-mono frac-le) auto
  hence ln (1 + frac x / real (nat ⌊x⌋)) ≤ ln (1 + 1 / 11)
    using x by (subst ln-le-cancel-iff) (auto intro!: add-pos-nonneg)
  also have ... = ln (12 / 11) by simp
  also have ... ≤ 1585 / 18216
    using ln-approx-bounds[of 12 / 11 1] by (simp add: eval-nat-numeral)
  also have 9 / pi ^ 2 ≤ 9 / 3.141592653588 ^ 2
    using pi-approx by (intro divide-left-mono power-mono mult-pos-pos) auto
  also have 1 + ... + 1585 / 18216 < 2
    by (simp add: power2-eq-square)
  finally show ?thesis by simp
 qed
 with ‹𝔐 x - ln x ≤ 25 / 18› show ?thesis by simp
qed

corollary mertens-first-theorem:
 fixes x :: real assumes x: x ≥ 1
 shows   |𝔐 x - ln x| < 2
 using mertens-bound-real'[of x] x by (simp add: abs-if)
```

5.2 Mertens' Second Theorem

Mertens' Second Theorem concerns the asymptotics of the Prime Harmonic Series, namely

$$\sum_{p \leq x} \frac{1}{p} = \ln \ln x + M + O\left(\frac{1}{\ln x}\right)$$

where $M \approx 0.261497$ is the Meissel–Mertens constant.

We define the constant in the following way:

```
definition meissel-mertens where
```

meissel-mertens = *1* − *ln* (*ln 2*) + *integral* {*2*..} (λ*t*. (𝔐 *t* − *ln t*) / (*t* ∗ *ln t* ^ *2*))

We will require the value of the integral $\int_a^\infty \frac{t}{\ln^2 t}\mathrm{d}t = \frac{1}{\ln a}$ as an upper bound on the remainder term:

lemma *integral-one-over-x-ln-x-squared*:
assumes *a*: (*a*::*real*) > *1*
shows *set-integrable lborel* {*a*<..} (λ*t*. *1* / (*t* ∗ *ln t* ^ *2*)) (**is** *?th1*)
and *set-lebesgue-integral lborel* {*a*<..} (λ*t*. *1* / (*t* ∗ *ln t* ^ *2*)) = *1* / *ln a* (**is** *?th2*)
and ((λ*t*. *1* / (*t* ∗ (*ln t*)²)) *has-integral 1* / *ln a*) {*a*<..} (**is** *?th3*)
proof −
have *cont*: *isCont* (λ*t*. *1* / (*t* ∗ (*ln t*)²)) *x* **if** *x* > *a* **for** *x*
using *that a* **by** (*auto intro*!: *continuous-intros*)
have *deriv*: ((λ*x*. −*1* / *ln x*) *has-real-derivative 1* / (*x* ∗ (*ln x*)²)) (*at x*) **if** *x* > *a* **for** *x*
using *that a* **by** (*auto intro*!: *derivative-eq-intros simp*: *power2-eq-square field-simps*)
have *lim1*: (((λ*x*. − *1* / *ln x*) ∘ *real-of-ereal*) ⟶ −(*1* / *ln a*)) (*at-right* (*ereal a*))
unfolding *ereal-tendsto-simps* **using** *a* **by** (*real-asymp simp*: *field-simps*)
have *lim2*: (((λ*x*. − *1* / *ln x*) ∘ *real-of-ereal*) ⟶ *0*) (*at-left* ∞)
unfolding *ereal-tendsto-simps* **using** *a* **by** (*real-asymp simp*: *field-simps*)

have *set-integrable lborel* (*einterval a* ∞) (λ*t*. *1* / (*t* ∗ *ln t* ^ *2*))
by (*rule interval-integral-FTC-nonneg*[*OF - deriv cont - lim1 lim2*]) (*use a* **in** *auto*)
thus *?th1* **by** *simp*
have *interval-lebesgue-integral lborel* (*ereal a*) ∞ (λ*t*. *1* / (*t* ∗ *ln t* ^ *2*)) = *0* − (−(*1* / *ln a*))
by (*rule interval-integral-FTC-nonneg*[*OF - deriv cont - lim1 lim2*]) (*use a* **in** *auto*)
thus *?th2* **by** (*simp add*: *interval-integral-to-infinity-eq*)

have ((λ*t*. *1* / (*t* ∗ *ln t* ^ *2*)) *has-integral*
set-lebesgue-integral lebesgue {*a*<..} (λ*t*. *1* / (*t* ∗ *ln t* ^ *2*))) {*a*<..}
using ‹*?th1*› **by** (*intro has-integral-set-lebesgue*)
(*auto simp*: *set-integrable-def integrable-completion*)
also have *set-lebesgue-integral lebesgue* {*a*<..} (λ*t*. *1* / (*t* ∗ *ln t* ^ *2*)) = *1* / *ln a*
using ‹*?th2*› **unfolding** *set-lebesgue-integral-def* **by** (*subst integral-completion*) *auto*
finally show *?th3* **.**
qed

We show that the integral in our definition of the Meissel–Mertens constant is well-defined and give an upper bound for its tails:

lemma
assumes *a* > (*1* :: *real*)
defines *r* ≡ (λ*t*. (𝔐 *t* − *ln t*) / (*t* ∗ *ln t* ^ *2*))

shows *integrable-meissel-mertens*: *set-integrable lborel* {*a*<..} *r*
and *meissel-mertens-integral-le*: *norm* (*integral* {*a*<..} *r*) ≤ *2* / *ln a*
proof −
have ∗: ((λ*t*. *2* ∗ (*1* / (*t* ∗ *ln t* ^ *2*))) *has-integral 2* ∗ (*1* / *ln a*)) {*a*<..}
using *assms* **by** (*intro has-integral-mult-right integral-one-over-x-ln-x-squared*)
auto
show *set-integrable lborel* {*a*<..} *r* **unfolding** *set-integrable-def*
proof (*rule Bochner-Integration.integrable-bound*[*OF - - AE-I2*])
have *integrable lborel* (λ*t*::*real*. *indicator* {*a*<..} *t* ∗ (*2* ∗ (*1* / (*t* ∗ *ln t* ^ *2*))))
using *integrable-mult-right*[*of 2*,
OF integral-one-over-x-ln-x-squared(*1*)[*of a*, *unfolded set-integrable-def*]]
assms
by (*simp add*: *algebra-simps*)
thus *integrable lborel* (λ*t*::*real*. *indicator* {*a*<..} *t* ∗*R* (*2* / (*t* ∗ *ln t* ^ *2*)))
by *simp*
fix *x* :: *real*
show *norm* (*indicat-real* {*a*<..} *x* ∗*R* *r x*) ≤
norm (*indicat-real* {*a*<..} *x* ∗*R* (*2* / (*x* ∗ *ln x* ^ *2*)))
proof (*cases x* > *a*)
case *True*
thus *?thesis*
unfolding *norm-scaleR norm-mult r-def norm-divide* **using** *mertens-first-theorem*[*of x*] *assms*
by (*intro mult-mono frac-le divide-nonneg-pos*) (*auto simp*: *indicator-def*)
qed (*auto simp*: *indicator-def*)
qed (*auto simp*: *r-def*)
hence *r integrable-on* {*a*<..}
by (*simp add*: *set-borel-integral-eq-integral*(*1*))
hence *norm* (*integral* {*a*<..} *r*) ≤ *integral* {*a*<..} (λ*x*. *2* ∗ (*1* / (*x* ∗ *ln x* ^ *2*)))
proof (*rule integral-norm-bound-integral*)
show (λ*x*. *2* ∗ (*1* / (*x* ∗ (*ln x*)^2))) *integrable-on* {*a*<..}
using ∗ **by** (*simp add*: *has-integral-iff*)
fix *x* **assume** *x* ∈ {*a*<..}
hence *norm* (*r x*) ≤ *2* / (*x* ∗ (*ln x*)^2)
unfolding *r-def norm-divide* **using** *mertens-first-theorem*[*of x*] *assms*
by (*intro mult-mono frac-le divide-nonneg-pos*) (*auto simp*: *indicator-def*)
thus *norm* (*r x*) ≤ *2*∗ (*1* / (*x* ∗ *ln x* ^ *2*)) **by** *simp*
qed
also have ... = *2* / *ln a*
using ∗ **by** (*simp add*: *has-integral-iff*)
finally show *norm* (*integral* {*a*<..} *r*) ≤ *2* / *ln a* **.**
qed

lemma *integrable-on-meissel-mertens*:
assumes *A* ⊆ {*1*..} *Inf A* > *1 A* ∈ *sets borel*
shows (λ*t*. (𝔐 *t* − *ln t*) / (*t* ∗ *ln t* ^ *2*)) *integrable-on A*
proof −
from *assms* **obtain** *x* **where** *x*: *1* < *x x* < *Inf A*
using *dense* **by** *blast*

```
  from assms have bdd-below A by (intro bdd-belowI[of - 1]) auto
  hence A ⊆ {Inf A..} by (auto simp: cInf-lower)
  also have ... ⊆ {x<..} using x by auto
  finally have ∗: A ⊆ {x<..} .
  have set-integrable lborel A (λt. (𝔐 t − ln t) / (t ∗ ln t ^ 2))
    by (rule set-integrable-subset[OF integrable-meissel-mertens[of x]]) (use x ∗
assms in auto)
  thus ?thesis by (simp add: set-borel-integral-eq-integral(1))
qed

lemma meissel-mertens-bounds: |meissel-mertens − 1 + ln (ln 2)| ≤ 2 / ln 2
proof −
  have ∗: {2..} − {2<..} = {2::real} by auto
  also have negligible ... by simp
  finally have integral {2..} (λt. (𝔐 t − ln t) / (t ∗ (ln t)^2)) =
                integral {2<..} (λt. (𝔐 t − ln t) / (t ∗ (ln t)^2))
    by (intro sym[OF integral-subset-negligible]) auto
  also have norm ... ≤ 2 / ln 2
    by (rule meissel-mertens-integral-le) auto
  finally show |meissel-mertens − 1 + ln (ln 2)| ≤ 2 / ln 2
    by (simp add: meissel-mertens-def)
qed
```

Finally, obtaining Mertens' second theorem from the first one is nothing but a routine summation by parts, followed by a use of the above bound:

```
theorem mertens-second-theorem:
  defines f ≡ prime-sum-upto (λp. 1 / p)
  shows   ⋀x. x ≥ 2 ⟹ |f x − ln (ln x) − meissel-mertens| ≤ 4 / ln x
    and   (λx. f x − ln (ln x) − meissel-mertens) ∈ O(λx. 1 / ln x)
proof −
  define r where r = (λt. (𝔐 t − ln t) / (t ∗ ln t ^ 2))

  {
    fix x :: real assume x: x > 2
    have ((λt. 𝔐 t ∗ (−1 / (t ∗ ln t ^ 2))) has-integral 𝔐 x ∗ (1 / ln x) − 𝔐 2
∗ (1 / ln 2) −
          (∑ n∈real −‘ {2<..x}. ind prime n ∗ (ln n / real n) ∗ (1 / ln n))) {2..x}
      unfolding primes-M-def prime-sum-upto-altdef1 using x
      by (intro partial-summation-strong[of {}])
         (auto intro!: continuous-intros derivative-eq-intros simp: power2-eq-square
               simp flip: has-real-derivative-iff-has-vector-derivative)
    also have 𝔐 x ∗ (1 / ln x) − 𝔐 2 ∗ (1 / ln 2) −
                (∑ n∈real −‘ {2<..x}. ind prime n ∗ (ln n / n) ∗ (1 / ln n)) =
              𝔐 x / ln x − (∑ n∈insert 2 (real −‘ {2<..x}). ind prime n ∗ (ln n /
n) ∗ (1 / ln n))
      (is - = - − ?S)
      by (subst sum.insert)
           (auto simp: primes-M-def finite-vimage-real-of-nat-greaterThanAtMost
eval-prime-sum-upto)
```

```
    also have ?S = f x
      unfolding f-def prime-sum-upto-altdef1 sum-upto-def using x
       by (intro sum.mono-neutral-cong-left) (auto simp: not-less numeral-2-eq-2
le-Suc-eq)
    finally have ((λt. -𝔐 t / (t * ln t ^ 2)) has-integral (𝔐 x / ln x - f x))
{2..x}
      by simp
    from has-integral-neg[OF this]
      have ((λt. 𝔐 t / (t * ln t ^ 2)) has-integral (f x - 𝔐 x / ln x)) {2..x} by
simp
    hence ((λt. 𝔐 t / (t * ln t ^ 2) - 1 / (t * ln t)) has-integral
          (f x - 𝔐 x / ln x - (ln (ln x) - ln (ln 2)))) {2..x} using x
      by (intro has-integral-diff fundamental-theorem-of-calculus)
        (auto simp flip: has-real-derivative-iff-has-vector-derivative
            intro!: derivative-eq-intros)
    also have ?this ⟷ (r has-integral (f x - 𝔐 x / ln x - (ln (ln x) - ln (ln
2)))) {2..x}
      by (intro has-integral-cong) (auto simp: r-def field-simps power2-eq-square)
    finally have ... .
  } note integral = this

  define R_𝔐 where R_𝔐 = (λx. 𝔐 x - ln x)
  have 𝔐: 𝔐 x = ln x + R_𝔐 x for x by (simp add: R_𝔐-def)
  define I where I = (λx. integral {x..} r)
  define C where C = (1 - ln (ln 2) + I 2)
  have C-altdef: C = meissel-mertens
    by (simp add: I-def r-def C-def meissel-mertens-def)

  show bound: |f x - ln (ln x) - meissel-mertens| ≤ 4 / ln x if x: x ≥ 2 for x
  proof (cases x = 2)
    case True
     hence |f x - ln (ln x) - meissel-mertens| = |1 / 2 - ln (ln 2) - meis-
sel-mertens|
      by (simp add: f-def eval-prime-sum-upto )
    also have ... ≤ 2 / ln 2 + 1 / 2
      using meissel-mertens-bounds by linarith
    also have ... ≤ 2 / ln 2 + 2 / ln 2 using ln2-le-25-over-36
      by (intro add-mono divide-left-mono) auto
    finally show ?thesis using True by simp
  next
    case False
    hence x: x > 2 using x by simp
   have integral {2..x} r + I x = integral ({2..x} ∪ {x..}) r unfolding I-def r-def
using x
      by (intro integral-Un [symmetric] integrable-on-meissel-mertens) (auto simp:
max-def r-def)
    also have {2..x} ∪ {x..} = {2..} using x by auto
    finally have *: integral {2..x} r = I 2 - I x unfolding I-def by simp
    have eq: f x - ln (ln x) - C = R_𝔐 x / ln x - I x
```

```
    using integral[OF x] x by (auto simp: C-def field-simps 𝔐 has-integral-iff ∗)
  also have |. . . | ≤ |R_𝔐 x / ln x| + norm (I x)
    unfolding real-norm-def by (rule abs-triangle-ineq4)
  also have |R_𝔐 x / ln x| ≤ 2 / |ln x|
    unfolding R_𝔐-def abs-divide using mertens-first-theorem[of x] x
    by (intro divide-right-mono) auto
  also have {x..} − {x<..} = {x} and {x<..} ⊆ {x..} by auto
  hence I x = integral {x<..} r unfolding I-def
    by (intro integral-subset-negligible [symmetric]) simp-all
  also have norm . . . ≤ 2 / ln x
    using meissel-mertens-integral-le[of x] x by (simp add: r-def)
  finally show |f x − ln (ln x) − meissel-mertens| ≤ 4 / ln x
    using x by (simp add: C-altdef)
qed

have (λx. f x − ln (ln x) − C) ∈ O(λx. 1 / ln x)
proof (intro landau-o.bigI[of 4] eventually-mono[OF eventually-ge-at-top[of 2]])
  fix x :: real assume x: x ≥ 2
  with bound[OF x] show norm (f x − ln (ln x) − C) ≤ 4 ∗ norm (1 / ln x)
    by (simp add: C-altdef)
qed (auto intro!: add-pos-nonneg)
thus (λx. f x − ln (ln x) − meissel-mertens) ∈ O(λx. 1 / ln x)
  by (simp add: C-altdef)
qed

corollary prime-harmonic-asymp-equiv: prime-sum-upto (λp. 1 / p) ∼[at-top] (λx.
ln (ln x))
proof −
  define f where f = prime-sum-upto (λp. 1 / p)
  have (λx. f x − ln (ln x) − meissel-mertens + meissel-mertens) ∈ o(λx. ln (ln
x))
    unfolding f-def
   by (rule sum-in-smallo[OF landau-o.big-small-trans[OF mertens-second-theorem(2)]])
real-asymp+
  hence (λx. f x − ln (ln x)) ∈ o(λx. ln (ln x))
    by simp
  thus ?thesis unfolding f-def
    by (rule smallo-imp-asymp-equiv)
qed
```

As a corollary, we get the divergence of the prime harmonic series.

```
corollary prime-harmonic-diverges: filterlim (prime-sum-upto (λp. 1 / p)) at-top
at-top
  using asymp-equiv-symI[OF prime-harmonic-asymp-equiv]
  by (rule asymp-equiv-at-top-transfer) real-asymp

end
```

6 Acknowledgements

Paulson was supported by the ERC Advanced Grant ALEXANDRIA (Project 742178) funded by the European Research Council at the University of Cambridge, UK.

References

[1] T. M. Apostol. *Introduction to Analytic Number Theory*. Undergraduate Texts in Mathematics. Springer-Verlag, 1976.

[2] J. Avigad, K. Donnelly, D. Gray, and P. Raff. A formally verified proof of the prime number theorem. *ACM Trans. Comput. Logic*, 9(1), Dec. 2007.

[3] M. Carneiro. Formalization of the prime number theorem and dirichlet's theorem. In *Proceedings of the 9th Conference on Intelligent Computer Mathematics (CICM 2016)*, pages 10–13, 2016.

[4] O. Forster. Analytic Number Theory (lecture notes). http://www.mathematik.uni-muenchen.de/~forster/v/ann/annth_all.pdf.

[5] J. Harrison. Formalizing an analytic proof of the Prime Number Theorem (dedicated to Mike Gordon on the occasion of his 60th birthday). *Journal of Automated Reasoning*, 43:243–261, 2009.

[6] D. Newman. *Analytic Number Theory*. Number 177 in Graduate Texts in Mathematics. Springer, 1998.