

Elementary Facts About the Distribution of Primes

Manuel Eberl

December 14, 2021

Abstract

This entry is a formalisation of Chapter 4 (and parts of Chapter 3) of Apostol's *Introduction to Analytic Number Theory*. The main topics that are addressed are properties of the distribution of prime numbers that can be shown in an elementary way (i.e. without the Prime Number Theorem), the various equivalent forms of the PNT (which imply each other in elementary ways), and consequences that follow from the PNT in elementary ways. The latter include bounds for the number of distinct prime factors of n , the divisor function $d(n)$, Euler's totient function $\varphi(n)$, and $\text{lcm}(1, \dots, n)$.

Contents

1	Auxiliary material	4
1.1	Various facts about Dirichlet series	8
1.2	Facts about prime-counting functions	10
1.3	Strengthening ‘Big-O’ bounds	11
2	Miscellaneous material	16
2.1	Generalised Dirichlet products	16
2.2	Legendre’s identity	20
2.3	A weighted sum of the Möbius μ function	24
3	The Prime ω function	25
4	The Primorial function	27
4.1	Definition and basic properties	27
4.2	An alternative view on primorials	29
4.3	Maximal compositeness of primorials	31
5	The LCM of the first n natural numbers	34
6	Shapiro’s Tauberian Theorem	37
6.1	Proof	37
6.2	Applications to the Chebyshev functions	44
7	Bounds on partial sums of the ζ function	46
8	The summatory Möbius μ function	56
9	Elementary bounds on $\pi(x)$ and p_n	67
9.1	Preliminary lemmas	68
9.2	Lower bound for $\pi(x)$	69
9.3	Upper bound for $\vartheta(x)$	72
9.4	Upper bound for $\pi(x)$	75
9.5	Bounds for p_n	79
10	The asymptotics of the summatory divisor σ function	81
10.1	Case 1: $\alpha = 1$	82
10.2	Case 2: $\alpha > 0, \alpha \neq 1$	84
10.3	Case 3: $\alpha < 0$	86
11	Selberg’s asymptotic formula	89

12	Consequences of the Prime Number Theorem	95
12.1	Statement and alternative forms of the PNT	95
12.2	Existence of primes in intervals	98
12.3	The logarithm of the primorial	99
12.4	Consequences of the asymptotics of ψ and ϑ	101
12.5	Bounds on the prime ω function	103
12.6	Bounds on the divisor function	105
12.7	Mertens' Third Theorem	111
12.8	Bounds on Euler's totient function	116

1 Auxiliary material

theory *Prime-Distribution-Elementary-Library*

imports

Zeta-Function.Zeta-Function

Prime-Number-Theorem.Prime-Counting-Functions

Stirling-Formula.Stirling-Formula

begin

lemma *divisor-count-pos* [intro]: $n > 0 \implies \text{divisor-count } n > 0$

by (*auto simp: divisor-count-def intro!: Nat.gr0I*)

lemma *divisor-count-eq-0-iff* [simp]: $\text{divisor-count } n = 0 \iff n = 0$

by (*cases n = 0*) *auto*

lemma *divisor-count-pos-iff* [simp]: $\text{divisor-count } n > 0 \iff n > 0$

by (*cases n = 0*) *auto*

lemma *smallest-prime-beyond-eval*:

prime n \implies smallest-prime-beyond n = n

\neg prime n \implies smallest-prime-beyond n = smallest-prime-beyond (Suc n)

proof –

assume *prime n*

thus *smallest-prime-beyond n = n*

by (*rule smallest-prime-beyond-eq*) *auto*

next

assume *\neg prime n*

show *smallest-prime-beyond n = smallest-prime-beyond (Suc n)*

proof (*rule antisym*)

show *smallest-prime-beyond n \leq smallest-prime-beyond (Suc n)*

by (*rule smallest-prime-beyond-smallest*)

(*auto intro: order.trans[OF - smallest-prime-beyond-le]*)

next

have *smallest-prime-beyond n \neq n*

using *prime-smallest-prime-beyond[of n] $\langle \neg$ prime n \rangle* **by** *metis*

hence *smallest-prime-beyond n $>$ n*

using *smallest-prime-beyond-le[of n]* **by** *linarith*

thus *smallest-prime-beyond n \geq smallest-prime-beyond (Suc n)*

by (*intro smallest-prime-beyond-smallest*) *auto*

qed

qed

lemma *nth-prime-numeral*:

nth-prime (numeral n) = smallest-prime-beyond (Suc (nth-prime (pred-numeral n)))

by (*subst nth-prime-Suc[symmetric]*) *auto*

lemmas *nth-prime-eval = smallest-prime-beyond-eval nth-prime-Suc nth-prime-numeral*

lemma *nth-prime-1* [*simp*]: *nth-prime* (Suc 0) = 3
by (*simp add: nth-prime-eval*)

lemma *nth-prime-2* [*simp*]: *nth-prime* 2 = 5
by (*simp add: nth-prime-eval*)

lemma *nth-prime-3* [*simp*]: *nth-prime* 3 = 7
by (*simp add: nth-prime-eval*)

lemma *strict-mono-sequence-partition*:

assumes *strict-mono* (*f* :: nat \Rightarrow 'a :: {*linorder*, *no-top*})

assumes $x \geq f 0$

assumes *filterlim* *f at-top at-top*

shows $\exists k. x \in \{f k..<f (Suc k)\}$

proof –

define *k* **where** $k = (LEAST k. f (Suc k) > x)$

{

obtain *n* **where** $x \leq f n$

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 $\exists n. f (Suc n) > x$ **by** *auto*

}

from *LeastI-ex[OF this]* **have** $x < f (Suc k)$

by (*simp add: k-def*)

moreover have $f k \leq x$

proof (*cases k*)

case (*Suc k'*)

have $k \leq k'$ **if** $f (Suc k') > x$

using *that unfolding k-def* **by** (*rule Least-le*)

with *Suc* **show** $f k \leq x$ **by** (*cases f k \leq x*) (*auto simp: not-le*)

qed (*use assms in auto*)

ultimately show *?thesis* **by** *auto*

qed

lemma *nth-prime-partition*:

assumes $x \geq 2$

shows $\exists k. x \in \{nth\text{-prime } k..<nth\text{-prime } (Suc k)\}$

using *strict-mono-sequence-partition[OF strict-mono-nth-prime, of x]* *assms nth-prime-at-top*

by *simp*

lemma *nth-prime-partition'*:

assumes $x \geq 2$

shows $\exists k. x \in \{real (nth\text{-prime } k)..<real (nth\text{-prime } (Suc k))\}$

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 $n > \text{nth-prime } k \ n < \text{nth-prime } (\text{Suc } k)$
shows $\neg \text{prime } n$
using *assms* **by** (*metis Suc-leI not-le nth-prime-Suc smallest-prime-beyond-smallest*)

lemma *nth-prime-partition'*:
includes *prime-counting-notation*
assumes $x \geq (2 :: \text{real})$
shows $x \in \{\text{real } (\text{nth-prime } (\text{nat } \lfloor \pi x \rfloor - 1)) .. < \text{real } (\text{nth-prime } (\text{nat } \lfloor \pi x \rfloor))\}$
proof –
obtain n **where** $n: x \in \{\text{nth-prime } n .. < \text{nth-prime } (\text{Suc } n)\}$
using *nth-prime-partition'* **assms** **by** *auto*
have $\pi (\text{nth-prime } n) = \pi x$
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 $\text{real } n = \pi x - 1$
by *simp*
hence *n-eq*: $n = \text{nat } \lfloor \pi x \rfloor - 1 \ \text{Suc } n = \text{nat } \lfloor \pi x \rfloor$
by *linarith+*
with n **show** *?thesis*
by *simp*
qed

lemma *asympt-equivD-strong*:
assumes $f \sim [F] g$ *eventually* $(\lambda x. f x \neq 0 \vee g x \neq 0) F$
shows $((\lambda x. f x / g x) \longrightarrow 1) F$
proof –
from *assms(1)* **have** $((\lambda x. \text{if } f x = 0 \wedge g x = 0 \text{ then } 1 \text{ else } f x / g x) \longrightarrow 1) F$
by (*rule asympt-equivD*)
also **have** *?this* \longleftrightarrow *?thesis*
by (*intro filterlim-cong eventually-mono[OF assms(2)]*) *auto*
finally **show** *?thesis* .
qed

lemma *hurwitz-zeta-shift*:
fixes $s :: \text{complex}$
assumes $a > 0$ **and** $s \neq 1$
shows $\text{hurwitz-zeta } (a + \text{real } n) s = \text{hurwitz-zeta } a s - (\sum_{k < n. (a + \text{real } k)^{\text{powr } -s})$
proof (*rule analytic-continuation-open* [**where** $f = \lambda s. \text{hurwitz-zeta } (a + \text{real } n) s$])
fix s **assume** $s: s \in \{s. \text{Re } s > 1\}$
have $(\lambda k. (a + \text{of-nat } (k + n))^{\text{powr } -s}) \text{ sums } \text{hurwitz-zeta } (a + \text{real } n) s$
using *sums-hurwitz-zeta* [*of a + real n s*] s **assms** **by** (*simp add: add-ac*)
moreover **have** $(\lambda k. (a + \text{of-nat } k)^{\text{powr } -s}) \text{ sums } \text{hurwitz-zeta } a s$
using *sums-hurwitz-zeta* [*of a s*] s **assms** **by** (*simp add: add-ac*)
hence $(\lambda k. (a + \text{of-nat } (k + n))^{\text{powr } -s}) \text{ sums}$

$(\text{hurwitz-zeta } a \ s - (\sum k < n. (a + \text{of-nat } k) \text{ powr } -s))$
by (*rule sums-split-initial-segment*)
ultimately show $\text{hurwitz-zeta } (a + \text{real } n) \ s = \text{hurwitz-zeta } a \ s - (\sum k < n. (a + \text{real } k) \text{ powr } -s)$
by (*simp add: sums-iff*)
next
show *connected* $(-\{1 :: \text{complex}\})$
by (*rule connected-punctured-universe*) *auto*
qed (*use assms in ‹auto intro!: holomorphic-intros open-halfspace-Re-gt exI[of - 2]›*)

lemma *pbernpoly-bigo*: $pbernpoly \ n \in O(\lambda-. \ 1)$
proof –
from *bounded-pbernpoly[of n]* **obtain** *c* **where** $\bigwedge x. \text{norm } (pbernpoly \ n \ x) \leq c$
by *auto*
thus *?thesis* **by** (*intro bigoI[of - c]*) *auto*
qed

lemma *harm-le*: $n \geq 1 \implies \text{harm } n \leq \ln \ n + 1$
using *euler-mascheroni-sequence-decreasing[of 1 n]*
by (*simp add: harm-expand*)

lemma *sum-upto-1* [*simp*]: $\text{sum-upto } f \ 1 = f \ 1$
proof –
have $\{0 <.. \text{Suc } 0\} = \{1\}$ **by** *auto*
thus *?thesis* **by** (*simp add: sum-upto-altdef*)
qed

lemma *sum-upto-cong'* [*cong*]:
 $(\bigwedge n. \ n > 0 \implies \text{real } n \leq x \implies f \ n = f' \ n) \implies x = x' \implies \text{sum-upto } f \ x = \text{sum-upto } f' \ x'$
unfolding *sum-upto-def* **by** (*intro sum.cong*) *auto*

lemma *finite-primes-le*: $\text{finite } \{p. \ \text{prime } p \wedge \text{real } p \leq x\}$
by (*rule finite-subset[of - \{..nat [x]\}]*) (*auto simp: le-nat-iff le-floor-iff*)

lemma *frequently-filtermap*: $\text{frequently } P \ (\text{filtermap } f \ F) = \text{frequently } (\lambda n. \ P \ (f \ n)) \ F$
by (*auto simp: frequently-def eventually-filtermap*)

lemma *frequently-mono-filter*: $\text{frequently } P \ F \implies F \leq F' \implies \text{frequently } P \ F'$
using *filter-leD*[*of F F' \lambda x. \neg P x*] **by** (*auto simp: frequently-def*)

lemma *\pi-at-top*: *filterlim primes-pi at-top at-top*
unfolding *filterlim-at-top*
proof *safe*
fix *C :: real*
define *x0* **where** $x0 = \text{real } (\text{nth-prime } (\text{nat } \lceil \text{max } 0 \ C \rceil))$

show *eventually* $(\lambda x. \text{primes-pi } x \geq C)$ *at-top*
using *eventually-ge-at-top*
proof *eventually-elim*
fix x **assume** $x \geq x0$
have $C \leq \text{real } (\text{nat } \lceil \text{max } 0 \ C \rceil + 1)$ **by** *linarith*
also have $\text{real } (\text{nat } \lceil \text{max } 0 \ C \rceil + 1) = \text{primes-pi } x0$
unfolding *x0-def* **by** *simp*
also have $\dots \leq \text{primes-pi } x$ **by** (*rule* π -*mono*) *fact*
finally show $\text{primes-pi } x \geq C$.
qed
qed

lemma *sum-upto-ln-stirling-weak-bigo*: $(\lambda x. \text{sum-upto } \ln x - x * \ln x + x) \in O(\ln)$
proof –
let $?f = \lambda x. x * \ln x - x + \ln (2 * \text{pi} * x) / 2$
have $\ln (\text{fact } n) - (n * \ln n - n + \ln (2 * \text{pi} * n) / 2) \in \{0..1/(12*n)\}$ **if** $n > 0$ **for** $n :: \text{nat}$
using *ln-fact-bounds*[*OF that*] **by** (*auto simp: algebra-simps*)
hence $(\lambda n. \ln (\text{fact } n) - ?f n) \in O(\lambda n. 1 / \text{real } n)$
by (*intro bigoI*[*of - 1/12*] *eventually-mono*[*OF eventually-gt-at-top*[*of 0*]]) *auto*
hence $(\lambda x. \ln (\text{fact } (\text{nat } \lfloor x \rfloor)) - ?f (\text{nat } \lfloor x \rfloor)) \in O(\lambda x. 1 / \text{real } (\text{nat } \lfloor x \rfloor))$
by (*rule landau-o.big.compose*)
(intro filterlim-compose[*OF filterlim-nat-sequentially*] *filterlim-floor-sequentially*)
also have $(\lambda x. 1 / \text{real } (\text{nat } \lfloor x \rfloor)) \in O(\lambda x::\text{real}. \ln x)$ **by** *real-asymp*
finally have $(\lambda x. \ln (\text{fact } (\text{nat } \lfloor x \rfloor)) - ?f (\text{nat } \lfloor x \rfloor) + (?f (\text{nat } \lfloor x \rfloor) - ?f x)) \in O(\lambda x. \ln x)$
by (*rule sum-in-bigo*) *real-asymp*
hence $(\lambda x. \ln (\text{fact } (\text{nat } \lfloor x \rfloor)) - ?f x) \in O(\lambda x. \ln x)$
by (*simp add: algebra-simps*)
hence $(\lambda x. \ln (\text{fact } (\text{nat } \lfloor x \rfloor)) - ?f x + \ln (2 * \text{pi} * x) / 2) \in O(\lambda x. \ln x)$
by (*rule sum-in-bigo*) *real-asymp*
thus *?thesis* **by** (*simp add: sum-upto-ln-conv-ln-fact algebra-simps*)
qed

1.1 Various facts about Dirichlet series

lemma *fds-mangoldt'*:

$$\text{fds mangoldt} = \text{fds-zeta} * \text{fds-deriv } (\text{fds moebius-mu})$$

proof –

$$\text{have } \text{fds mangoldt} = (\text{fds moebius-mu} * \text{fds } (\lambda n. \text{of-real } (\ln (\text{real } n)) :: 'a))$$

$$(\text{is } - = ?f) \text{ by } (\text{subst } \text{fds-mangoldt}) \text{ auto}$$

$$\text{also have } \dots = \text{fds-zeta} * \text{fds-deriv } (\text{fds moebius-mu})$$

proof (*intro fds-eqI*)

$$\text{fix } n :: \text{nat} \text{ assume } n: n > 0$$

$$\text{have } \text{fds-nth } ?f n = (\sum d \mid d \text{ dvd } n. \text{moebius-mu } d * \text{of-real } (\ln (\text{real } (n \text{ div } d))))$$

$$\text{by } (\text{auto simp: } \text{fds-eq-iff } \text{fds-nth-mult } \text{dirichlet-prod-def})$$

$$\text{also have } \dots = (\sum d \mid d \text{ dvd } n. \text{moebius-mu } d * \text{of-real } (\ln (\text{real } n / \text{real } d)))$$

$$\text{by } (\text{intro sum.cong}) (\text{auto elim!: } \text{dvdE } \text{simp: } \text{ln-mult } \text{split: if-splits})$$

also have ... = $(\sum d \mid d \text{ dvd } n. \text{moebius-mu } d * \text{of-real } (\ln n - \ln d))$
using n **by** $(\text{intro sum.cong refl}) (\text{subst ln-div, auto elim! : dvdE})$
also have ... = $\text{of-real } (\ln n) * (\sum d \mid d \text{ dvd } n. \text{moebius-mu } d) -$
 $(\sum d \mid d \text{ dvd } n. \text{of-real } (\ln d) * \text{moebius-mu } d)$
by $(\text{simp add: sum-subtractf sum-distrib-left sum-distrib-right algebra-simps})$
also have $\text{of-real } (\ln n) * (\sum d \mid d \text{ dvd } n. \text{moebius-mu } d) = 0$
by $(\text{subst sum-moebius-mu-divisors}^\wedge \text{ auto})$
finally show $\text{fds-nth } ?f n = \text{fds-nth } (\text{fds-zeta} * \text{fds-deriv } (\text{fds moebius-mu})) :: 'a$
 $\text{fds} n$
by $(\text{simp add: fds-nth-mult dirichlet-prod-altdef1 fds-nth-deriv sum-negf scaleR-conv-of-real})$
qed
finally show $?thesis$.
qed

lemma $\text{sum-upto-divisor-sum1}$:
 $\text{sum-upto } (\lambda n. \sum d \mid d \text{ dvd } n. f d :: \text{real}) x = \text{sum-upto } (\lambda n. f n * \text{floor } (x / n))$
 x
proof –
have $\text{sum-upto } (\lambda n. \sum d \mid d \text{ dvd } n. f d :: \text{real}) x =$
 $\text{sum-upto } (\lambda n. f n * \text{real } (\text{nat } (\text{floor } (x / n)))) x$
using $\text{sum-upto-dirichlet-prod}[of f \lambda-. 1 x]$
by $(\text{simp add: dirichlet-prod-def sum-upto-altdef})$
also have ... = $\text{sum-upto } (\lambda n. f n * \text{floor } (x / n)) x$
unfolding sum-upto-def **by** $(\text{intro sum.cong}) \text{ auto}$
finally show $?thesis$.
qed

lemma $\text{sum-upto-divisor-sum2}$:
 $\text{sum-upto } (\lambda n. \sum d \mid d \text{ dvd } n. f d :: \text{real}) x = \text{sum-upto } (\lambda n. \text{sum-upto } f (x / n))$
 x
using $\text{sum-upto-dirichlet-prod}[of \lambda-. 1 f x]$ **by** $(\text{simp add: dirichlet-prod-altdef1})$

lemma $\text{sum-upto-moebius-times-floor-linear}$:
 $\text{sum-upto } (\lambda n. \text{moebius-mu } n * \lfloor x / \text{real } n \rfloor) x = (\text{if } x \geq 1 \text{ then } 1 \text{ else } 0)$
proof –
have $\text{real-of-int } (\text{sum-upto } (\lambda n. \text{moebius-mu } n * \lfloor x / \text{real } n \rfloor) x) =$
 $\text{sum-upto } (\lambda n. \text{moebius-mu } n * \text{of-int } \lfloor x / \text{real } n \rfloor) x$
by $(\text{simp add: sum-upto-def})$
also have ... = $\text{sum-upto } (\lambda n. \sum d \mid d \text{ dvd } n. \text{moebius-mu } d :: \text{real}) x$
using $\text{sum-upto-divisor-sum1}[of \text{moebius-mu } x]$ **by** auto
also have ... = $\text{sum-upto } (\lambda n. \text{if } n = 1 \text{ then } 1 \text{ else } 0) x$
by $(\text{intro sum-upto-cong sum-moebius-mu-divisors' refl})$
also have ... = $\text{real-of-int } (\text{if } x \geq 1 \text{ then } 1 \text{ else } 0)$
by $(\text{auto simp: sum-upto-def})$
finally show $?thesis$ **unfolding** of-int-eq-iff .
qed

lemma $\text{ln-fact-conv-sum-mangoldt}$:
 $\text{sum-upto } (\lambda n. \text{mangoldt } n * \lfloor x / \text{real } n \rfloor) x = \ln (\text{fact } (\text{nat } \lfloor x \rfloor))$

proof –
have $sum\text{-upto } (\lambda n. mangoldt\ n * of\text{-int } \lfloor x / real\ n \rfloor) x =$
 $sum\text{-upto } (\lambda n. \sum d \mid d\ dvd\ n. mangoldt\ d :: real) x$
using $sum\text{-upto-divisor-sum1}$ [of $mangoldt\ x$] **by** $auto$
also have $\dots = sum\text{-upto } (\lambda n. of\text{-real } (ln\ (real\ n))) x$
by ($intro\ sum\text{-upto-cong\ mangoldt-sum\ refl$) $auto$
also have $\dots = (\sum n \in \{0 < .. nat\ \lfloor x \rfloor\}. ln\ n)$
by ($simp\ add: sum\text{-upto-altdef}$)
also have $\dots = ln\ (\prod \{0 < .. nat\ \lfloor x \rfloor\})$
unfolding $of\text{-nat-prod}$ **by** ($subst\ ln\text{-prod}$) $auto$
also have $\{0 < .. nat\ \lfloor x \rfloor\} = \{1 .. nat\ \lfloor x \rfloor\}$ **by** $auto$
also have $\prod \dots = fact\ (nat\ \lfloor x \rfloor)$
by ($simp\ add: fact\text{-prod}$)
finally show $?thesis$ **by** $simp$
qed

1.2 Facts about prime-counting functions

lemma $abs\text{-}\pi$ [$simp$]: $|primes\text{-}\pi\ x| = primes\text{-}\pi\ x$
by ($subst\ abs\text{-of-nonneg}$) $auto$

lemma $\pi\text{-less-self}$:
includes $prime\text{-counting-notation}$
assumes $x > 0$
shows $\pi\ x < x$

proof –
have $\pi\ x \leq (\sum n \in \{1 < .. nat\ \lfloor x \rfloor\}. 1)$
unfolding $\pi\text{-def}\ prime\text{-sum-upto-altdef2}$ **by** ($intro\ sum\text{-mono2}$) ($auto\ dest:$
 $prime\text{-gt-1-nat}$)
also have $\dots = real\ (nat\ \lfloor x \rfloor - 1)$
using $assms$ **by** $simp$
also have $\dots < x$ **using** $assms$ **by** $linarith$
finally show $?thesis$.
qed

lemma $\pi\text{-le-self}'$:
includes $prime\text{-counting-notation}$
assumes $x \geq 1$
shows $\pi\ x \leq x - 1$

proof –
have $\pi\ x \leq (\sum n \in \{1 < .. nat\ \lfloor x \rfloor\}. 1)$
unfolding $\pi\text{-def}\ prime\text{-sum-upto-altdef2}$ **by** ($intro\ sum\text{-mono2}$) ($auto\ dest:$
 $prime\text{-gt-1-nat}$)
also have $\dots = real\ (nat\ \lfloor x \rfloor - 1)$
using $assms$ **by** $simp$
also have $\dots \leq x - 1$ **using** $assms$ **by** $linarith$
finally show $?thesis$.
qed

lemma π -le-self:
includes *prime-counting-notation*
assumes $x \geq 0$
shows $\pi x \leq x$
using π -less-self[*of x*] *assms* **by** (*cases x = 0*) *auto*

1.3 Strengthening ‘Big-O’ bounds

The following two statements are crucial: They allow us to strengthen a ‘Big-O’ statement for $n \rightarrow \infty$ or $x \rightarrow \infty$ to a bound for *all* $n \geq n_0$ or all $x \geq x_0$ under some mild conditions.

This allows us to use all the machinery of asymptotics in Isabelle and still get a bound that is applicable over the full domain of the function in the end. This is important because Newman often shows that $f(x) \in O(g(x))$ and then writes

$$\sum_{n \leq x} f\left(\frac{x}{n}\right) = \sum_{n \leq x} O\left(g\left(\frac{x}{n}\right)\right)$$

which is not easy to justify otherwise.

lemma *natfun-bigoE*:
fixes $f :: \text{nat} \Rightarrow -$
assumes *bigo*: $f \in O(g)$ **and** *nz*: $\bigwedge n. n \geq n_0 \implies g\ n \neq 0$
obtains c **where** $c > 0 \bigwedge n. n \geq n_0 \implies \text{norm } (f\ n) \leq c * \text{norm } (g\ n)$
proof –
from *bigo* **obtain** c **where** $c: c > 0$ *eventually* $(\lambda n. \text{norm } (f\ n) \leq c * \text{norm } (g\ n))$ *at-top*
by (*auto elim: landau-o.bigoE*)
then obtain n_0' **where** $n_0': \bigwedge n. n \geq n_0' \implies \text{norm } (f\ n) \leq c * \text{norm } (g\ n)$
by (*auto simp: eventually-at-top-linorder*)
define c' **where** $c' = \text{Max } ((\lambda n. \text{norm } (f\ n) / \text{norm } (g\ n))$ ‘(*insert n_0 {n_0..<n_0'}*)’)
have $\text{norm } (f\ n) \leq \text{max } 1 (\text{max } c\ c') * \text{norm } (g\ n)$ **if** $n \geq n_0$ **for** n
proof (*cases n ≥ n_0'*)
case *False*
with that **have** $\text{norm } (f\ n) / \text{norm } (g\ n) \leq c'$
unfolding c' -*def* **by** (*intro Max.coboundedI*) *auto*
also have $\dots \leq \text{max } 1 (\text{max } c\ c')$ **by** *simp*
finally show *?thesis* **using** *nz[of n]* **that** **by** (*simp add: field-simps*)
next
case *True*
hence $\text{norm } (f\ n) \leq c * \text{norm } (g\ n)$ **by** (*rule n_0'*)
also have $\dots \leq \text{max } 1 (\text{max } c\ c') * \text{norm } (g\ n)$
by (*intro mult-right-mono*) *auto*
finally show *?thesis* .
qed
with that[*of max 1 (max c c')*] **show** *?thesis* **by** *auto*
qed

lemma *bigoE-bounded-real-fun*:

fixes $f\ g :: \text{real} \Rightarrow \text{real}$
assumes $f \in O(g)$
assumes $\bigwedge x. x \geq x0 \implies |g\ x| \geq cg\ cg > 0$
assumes $\bigwedge b. b \geq x0 \implies \text{bounded}\ (f'\ \{x0..b\})$
shows $\exists c > 0. \forall x \geq x0. |f\ x| \leq c * |g\ x|$
proof –
from *assms(1)* **obtain** c **where** $c: c > 0$ *eventually* $(\lambda x. |f\ x| \leq c * |g\ x|)$ *at-top*
by *(elim landau-o.bigE)* *auto*
then obtain b **where** $b: \bigwedge x. x \geq b \implies |f\ x| \leq c * |g\ x|$
by *(auto simp: eventually-at-top-linorder)*
have *bounded* $(f'\ \{x0..max\ x0\ b\})$ **by** *(intro assms)* *auto*
then obtain C **where** $C: \bigwedge x. x \in \{x0..max\ x0\ b\} \implies |f\ x| \leq C$
unfolding *bounded-iff* **by** *fastforce*

define c' **where** $c' = max\ c\ (C / cg)$
have $|f\ x| \leq c' * |g\ x|$ **if** $x \geq x0$ **for** x
proof *(cases* $x \geq b$ *)*
case *False*
then have $|f\ x| \leq C$
using C **that by** *auto*
with *False* **have** $|f\ x| / |g\ x| \leq C / cg$
by *(meson abs-ge-zero assms frac-le landau-omega.R-trans that)*
also have $\dots \leq c'$ **by** *(simp add: c'-def)*
finally show $|f\ x| \leq c' * |g\ x|$
using that *False assms(2)[of x] assms(3)* **by** *(auto simp add: divide-simps split: if-splits)*
next
case *True*
hence $|f\ x| \leq c * |g\ x|$ **by** *(intro b)* *auto*
also have $\dots \leq c' * |g\ x|$ **by** *(intro mult-right-mono)* *(auto simp: c'-def)*
finally show *?thesis* .
qed
moreover from *c(1)* **have** $c' > 0$ **by** *(auto simp: c'-def)*
ultimately show *?thesis* **by** *blast*
qed

lemma *sum-upto-asymptotics-lift-nat-real-aux:*

fixes $f :: \text{nat} \Rightarrow \text{real}$ **and** $g :: \text{real} \Rightarrow \text{real}$
assumes *bigO*: $(\lambda n. (\sum_{k=1..n} f\ k) - g\ (\text{real } n)) \in O(\lambda n. h\ (\text{real } n))$
assumes *g-bigO-self*: $(\lambda n. g\ (\text{real } n) - g\ (\text{real } (\text{Suc } n))) \in O(\lambda n. h\ (\text{real } n))$
assumes *h-bigO-self*: $(\lambda n. h\ (\text{real } n)) \in O(\lambda n. h\ (\text{real } (\text{Suc } n)))$
assumes *h-pos*: $\bigwedge x. x \geq 1 \implies h\ x > 0$
assumes *mono-g*: $\text{mono-on } g\ \{1..\} \vee \text{mono-on } (\lambda x. -g\ x)\ \{1..\}$
assumes *mono-h*: $\text{mono-on } h\ \{1..\} \vee \text{mono-on } (\lambda x. -h\ x)\ \{1..\}$
shows $\exists c > 0. \forall x \geq 1. \text{sum-upto } f\ x - g\ x \leq c * h\ x$
proof –
have *h-nz*: $h\ (\text{real } n) \neq 0$ **if** $n \geq 1$ **for** n
using *h-pos[of n]* **that by** *simp*

```

from natfun-bigoE[OF bigo h-nz] obtain c1 where
  c1: c1 > 0  $\wedge$  n. n  $\geq$  1  $\implies$  norm (( $\sum$  k=1..n. f k) - g (real n))  $\leq$  c1 * norm
(h (real n))
by auto
from natfun-bigoE[OF g-bigo-self h-nz] obtain c2 where
  c2: c2 > 0  $\wedge$  n. n  $\geq$  1  $\implies$  norm (g (real n) - g (real (Suc n)))  $\leq$  c2 * norm
(h (real n))
by auto
from natfun-bigoE[OF h-bigo-self h-nz] obtain c3 where
  c3: c3 > 0  $\wedge$  n. n  $\geq$  1  $\implies$  norm (h (real n))  $\leq$  c3 * norm (h (real (Suc n)))
by auto

{
  fix x :: real assume x: x  $\geq$  1
  define n where n = nat  $\lfloor$ x $\rfloor$ 
  from x have n: n  $\geq$  1 unfolding n-def by linarith

  have ( $\sum$  k = 1..n. f k) - g x  $\leq$  (c1 + c2) * h (real n) using mono-g
proof
  assume mono: mono-on ( $\lambda$ x. -g x) {1..}
  from x have x  $\leq$  real (Suc n)
  unfolding n-def by linarith
  hence ( $\sum$  k=1..n. f k) - g x  $\leq$  ( $\sum$  k=1..n. f k) - g n + (g n - g (Suc n))
  using mono-onD[OF mono, of x real (Suc n)] x by auto
  also have ...  $\leq$  norm (( $\sum$  k=1..n. f k) - g n) + norm (g n - g (Suc n))
  by simp
  also have ...  $\leq$  c1 * norm (h n) + c2 * norm (h n)
  using n by (intro add-mono c1 c2) auto
  also have ... = (c1 + c2) * h n
  using h-pos[of real n] n by (simp add: algebra-simps)
  finally show ?thesis .

  next
  assume mono: mono-on g {1..}
  have ( $\sum$  k=1..n. f k) - g x  $\leq$  ( $\sum$  k=1..n. f k) - g n
  using x by (intro diff-mono mono-onD[OF mono]) (auto simp: n-def)
  also have ...  $\leq$  c1 * h (real n)
  using c1(2)[of n] n h-pos[of n] by simp
  also have ...  $\leq$  (c1 + c2) * h (real n)
  using c2 h-pos[of n] n by (intro mult-right-mono) auto
  finally show ?thesis .

  qed
  also have (c1 + c2) * h (real n)  $\leq$  (c1 + c2) * (1 + c3) * h x
  using mono-h

  proof
  assume mono: mono-on ( $\lambda$ x. -h x) {1..}
  have (c1 + c2) * h (real n)  $\leq$  (c1 + c2) * (c3 * h (real (Suc n)))
  using c3(2)[of n] n h-pos[of n] h-pos[of Suc n] c1(1) c2(1)
  by (intro mult-left-mono) (auto)
  also have ... = (c1 + c2) * c3 * h (real (Suc n))

```

by (*simp add: mult-ac*)
 also have $\dots \leq (c1 + c2) * (1 + c3) * h \text{ (real (Suc n))}$
 using $c1(1) c2(1) c3(1) h\text{-pos[of Suc n]}$ by (*intro mult-left-mono mult-right-mono*)
auto
 also from x have $x \leq \text{real (Suc n)}$
 unfolding $n\text{-def}$ by *linarith*
 hence $(c1 + c2) * (1 + c3) * h \text{ (real (Suc n))} \leq (c1 + c2) * (1 + c3) * h x$
 using $c1(1) c2(1) c3(1) \text{ mono-onD[OF mono, of x real (Suc n)] x}$
 by (*intro mult-left-mono*) (*auto simp: n-def*)
 finally show $(c1 + c2) * h \text{ (real n)} \leq (c1 + c2) * (1 + c3) * h x$.
 next
 assume $\text{mono: mono-on } h \{1..\}$
 have $(c1 + c2) * h \text{ (real n)} = 1 * ((c1 + c2) * h \text{ (real n)})$ by *simp*
 also have $\dots \leq (1 + c3) * ((c1 + c2) * h \text{ (real n)})$
 using $c1(1) c2(1) c3(1) h\text{-pos[of n] x n}$ by (*intro mult-right-mono*) *auto*
 also have $\dots = (1 + c3) * (c1 + c2) * h \text{ (real n)}$
 by (*simp add: mult-ac*)
 also have $\dots \leq (1 + c3) * (c1 + c2) * h x$
 using $x c1(1) c2(1) c3(1) h\text{-pos[of n] n}$
 by (*intro mult-left-mono mono-onD[OF mono]*) (*auto simp: n-def*)
 finally show $(c1 + c2) * h \text{ (real n)} \leq (c1 + c2) * (1 + c3) * h x$
 by (*simp add: mult-ac*)
 qed
 also have $(\sum k = 1..n. f k) = \text{sum-upto } f x$
 unfolding $\text{sum-upto-altdef } n\text{-def}$ by (*intro sum.cong*) *auto*
 finally have $\text{sum-upto } f x - g x \leq (c1 + c2) * (1 + c3) * h x$.
 }
 moreover have $(c1 + c2) * (1 + c3) > 0$
 using $c1(1) c2(1) c3(1)$ by (*intro mult-pos-pos add-pos-pos*) *auto*
 ultimately show *?thesis* by *blast*
 qed

lemma *sum-upto-asymptotics-lift-nat-real:*

fixes $f :: \text{nat} \Rightarrow \text{real}$ and $g :: \text{real} \Rightarrow \text{real}$
 assumes $\text{bigo: } (\lambda n. (\sum k=1..n. f k) - g \text{ (real n)}) \in O(\lambda n. h \text{ (real n)})$
 assumes $\text{g-bigo-self: } (\lambda n. g \text{ (real n)} - g \text{ (real (Suc n))}) \in O(\lambda n. h \text{ (real n)})$
 assumes $\text{h-bigo-self: } (\lambda n. h \text{ (real n)}) \in O(\lambda n. h \text{ (real (Suc n))})$
 assumes $\text{h-pos: } \bigwedge x. x \geq 1 \implies h x > 0$
 assumes $\text{mono-g: mono-on } g \{1..\} \vee \text{mono-on } (\lambda x. - g x) \{1..\}$
 assumes $\text{mono-h: mono-on } h \{1..\} \vee \text{mono-on } (\lambda x. - h x) \{1..\}$
 shows $\exists c > 0. \forall x \geq 1. |\text{sum-upto } f x - g x| \leq c * h x$
 proof -
 have $\exists c > 0. \forall x \geq 1. \text{sum-upto } f x - g x \leq c * h x$
 by (*intro sum-upto-asymptotics-lift-nat-real-aux assms*)
 then obtain $c1$ where $c1: c1 > 0 \bigwedge x. x \geq 1 \implies \text{sum-upto } f x - g x \leq c1 * h x$
 by *auto*

have $(\lambda n. -(g \text{ (real n)} - g \text{ (real (Suc n))})) \in O(\lambda n. h \text{ (real n)})$

by (*subst landau-o.big.uminus-in-iff*) *fact*
 also have $(\lambda n. -(g (\text{real } n) - g (\text{real } (\text{Suc } n)))) = (\lambda n. g (\text{real } (\text{Suc } n)) - g (\text{real } n))$
 by *simp*
 finally have $(\lambda n. g (\text{real } (\text{Suc } n)) - g (\text{real } n)) \in O(\lambda n. h (\text{real } n))$.
 moreover {
 have $(\lambda n. -((\sum k=1..n. f k) - g (\text{real } n))) \in O(\lambda n. h (\text{real } n))$
 by (*subst landau-o.big.uminus-in-iff*) *fact*
 also have $(\lambda n. -((\sum k=1..n. f k) - g (\text{real } n))) =$
 $(\lambda n. (\sum k=1..n. -f k) + g (\text{real } n))$ by (*simp add: sum-negf*)
 finally have $(\lambda n. (\sum k=1..n. -f k) + g (\text{real } n)) \in O(\lambda n. h (\text{real } n))$.
 }
 ultimately have $\exists c>0. \forall x \geq 1. \text{sum-upto } (\lambda n. -f n) x - (-g x) \leq c * h x$
 using *mono-g*
 by (*intro sum-upto-asymptotics-lift-nat-real-aux assms*) (*simp-all add: disj-commute*)
 then obtain *c2* where *c2*: $c2 > 0 \wedge x. x \geq 1 \implies \text{sum-upto } (\lambda n. -f n) x + g x \leq c2 * h x$
 by *auto*

{
 fix *x* :: *real* assume *x*: $x \geq 1$
 have $\text{sum-upto } f x - g x \leq \max c1 c2 * h x$
 using *h-pos[of x] x* by (*intro order.trans[OF c1(2)] mult-right-mono*) *auto*
 moreover have $\text{sum-upto } (\lambda n. -f n) x + g x \leq \max c1 c2 * h x$
 using *h-pos[of x] x* by (*intro order.trans[OF c2(2)] mult-right-mono*) *auto*
 hence $-(\text{sum-upto } f x - g x) \leq \max c1 c2 * h x$
 by (*simp add: sum-upto-def sum-negf*)
 ultimately have $|\text{sum-upto } f x - g x| \leq \max c1 c2 * h x$ by *linarith*
 }
 moreover from *c1(1) c2(1)* have $\max c1 c2 > 0$ by *simp*
 ultimately show *?thesis* by *blast*
 qed

lemma (in *factorial-semiring*) *primepow-divisors-induct* [*case-names zero unit factor*]:

assumes $P 0 \wedge x. \text{is-unit } x \implies P x$
 $\bigwedge p k x. \text{prime } p \implies k > 0 \implies \neg p \text{ dvd } x \implies P x \implies P (p \wedge k * x)$
 shows $P x$
proof –
 have *finite* (*prime-factors x*) by *simp*
 thus *?thesis*
proof (*induction prime-factors x arbitrary: x rule: finite-induct*)
 case *empty*
 hence *prime-factors x* = {} by *metis*
 hence *prime-factorization x* = {#} by *simp*
 thus *?case* using *assms(1,2)* by (*auto simp: prime-factorization-empty-iff*)
next
 case (*insert p A x*)
 define *k* where *k* = *multiplicity p x*

```

have k > 0 using insert.hyps
  by (auto simp: prime-factors-multiplicity k-def)
have p: p ∈ prime-factors x using insert.hyps by auto
from p have x ≠ 0 ¬is-unit p by (auto simp: in-prime-factors-iff)

from multiplicity-decompose'[OF this] obtain y where y: x = p ^ k * y ¬p
dvd y
  by (auto simp: k-def)
have prime-factorization x = replicate-mset k p + prime-factorization y
  using p ⟨k > 0⟩ y unfolding y
  by (subst prime-factorization-mult)
  (auto simp: prime-factorization-prime-power in-prime-factors-iff)
moreover from y p have p ∉ prime-factors y
  by (auto simp: in-prime-factors-iff)
ultimately have prime-factors y = prime-factors x - {p}
  by auto
also have ... = A
  using insert.hyps by auto
finally have P y using insert by auto
thus P x
  unfolding y using y ⟨k > 0⟩ p by (intro assms(3)) (auto simp: in-prime-factors-iff)
qed
qed
end

```

2 Miscellaneous material

```

theory More-Dirichlet-Misc
imports
  Prime-Distribution-Elementary-Library
  Prime-Number-Theorem.Prime-Counting-Functions
begin

```

2.1 Generalised Dirichlet products

```

definition dirichlet-prod' :: (nat ⇒ 'a :: comm-semiring-1) ⇒ (real ⇒ 'a) ⇒ real
⇒ 'a where
  dirichlet-prod' f g x = sum-upto (λm. f m * g (x / real m)) x

```

lemma *dirichlet-prod'-one-left*:

```

dirichlet-prod' (λn. if n = 1 then 1 else 0) f x = (if x ≥ 1 then f x else 0)
proof -
  have dirichlet-prod' (λn. if n = 1 then 1 else 0) f x =
    (∑ i | 0 < i ∧ real i ≤ x. (if i = Suc 0 then 1 else 0) * f (x / real i))
  by (simp add: dirichlet-prod'-def sum-upto-def)
  also have ... = (∑ i ∈ (if x ≥ 1 then {1::nat} else {}). f x)
  by (intro sum.mono-neutral-cong-right) (auto split: if-splits)
  also have ... = (if x ≥ 1 then f x else 0)

```


by *simp*
finally show *?thesis* .
qed

lemma *dirichlet-prod'-cong*:

assumes $\bigwedge n. n > 0 \implies \text{real } n \leq x \implies f\ n = f'\ n$

assumes $\bigwedge y. y \geq 1 \implies y \leq x \implies g\ y = g'\ y$

assumes $x = x'$

shows $\text{dirichlet-prod}'\ f\ g\ x = \text{dirichlet-prod}'\ f'\ g'\ x'$

unfolding *dirichlet-prod'-def*

by (*intro sum-upto-cong' assms, (subst assms | simp add: assms field-simps)*)+

lemma *dirichlet-prod'-assoc*:

$\text{dirichlet-prod}'\ f\ (\lambda y. \text{dirichlet-prod}'\ g\ h\ y)\ x = \text{dirichlet-prod}'\ (\text{dirichlet-prod}\ f\ g)\ h\ x$

proof –

have $\text{dirichlet-prod}'\ f\ (\lambda y. \text{dirichlet-prod}'\ g\ h\ y)\ x =$

$(\sum m \mid m > 0 \wedge \text{real } m \leq x. \sum n \mid n > 0 \wedge \text{real } n \leq x / m. f\ m * g\ n * h\ (x / (m * n)))$

by (*simp add: algebra-simps dirichlet-prod'-def dirichlet-prod-def sum-upto-def sum-distrib-left sum-distrib-right*)

also have $\dots = (\sum (m,n) \in (\text{SIGMA } m: \{m. m > 0 \wedge \text{real } m \leq x\}. \{n. n > 0 \wedge \text{real } n \leq x / m\}).$

$f\ m * g\ n * h\ (x / (m * n)))$

by (*subst sum.Sigma*) *auto*

also have $\dots = (\sum (mn, m) \in (\text{SIGMA } mn: \{mn. mn > 0 \wedge \text{real } mn \leq x\}. \{m. m \text{ dvd } mn\}).$

$f\ m * g\ (mn \text{ div } m) * h\ (x / mn))$

by (*rule sum.reindex-bij-witness[of - $\lambda(mn, m). (m, mn \text{ div } m)$ $\lambda(m, n). (m * n, m)$]*)

(*auto simp: case-prod-unfold field-simps dest: dvd-imp-le*)

also have $\dots = \text{dirichlet-prod}'\ (\text{dirichlet-prod}\ f\ g)\ h\ x$

by (*subst sum.Sigma [symmetric]*)

(*simp-all add: dirichlet-prod'-def dirichlet-prod-def sum-upto-def algebra-simps sum-distrib-left sum-distrib-right*)

finally show *?thesis* .

qed

lemma *dirichlet-prod'-inversion1*:

assumes $\forall x \geq 1. g\ x = \text{dirichlet-prod}'\ a\ f\ x\ x \geq 1$

$\text{dirichlet-prod}\ a\ \text{ainv} = (\lambda n. \text{if } n = 1 \text{ then } 1 \text{ else } 0)$

shows $f\ x = \text{dirichlet-prod}'\ \text{ainv}\ g\ x$

proof –

have $\text{dirichlet-prod}'\ \text{ainv}\ g\ x = \text{dirichlet-prod}'\ \text{ainv}\ (\text{dirichlet-prod}'\ a\ f)\ x$

using *assms* **by** (*intro dirichlet-prod'-cong*) *auto*

also have $\dots = \text{dirichlet-prod}'\ (\lambda n. \text{if } n = 1 \text{ then } 1 \text{ else } 0)\ f\ x$

using *assms* **by** (*simp add: dirichlet-prod'-assoc dirichlet-prod-commutes*)

also have $\dots = f x$
using *assms* **by** (*subst dirichlet-prod'-one-left*) *auto*
finally show *?thesis ..*
qed

lemma *dirichlet-prod'-inversion2*:
assumes $\forall x \geq 1. f x = \text{dirichlet-prod}' \text{ainv } g x x \geq 1$
 $\text{dirichlet-prod } a \text{ainv} = (\lambda n. \text{if } n = 1 \text{ then } 1 \text{ else } 0)$
shows $g x = \text{dirichlet-prod}' a f x$
proof –
have $\text{dirichlet-prod}' a f x = \text{dirichlet-prod}' a (\text{dirichlet-prod}' \text{ainv } g) x$
using *assms* **by** (*intro dirichlet-prod'-cong*) *auto*
also have $\dots = \text{dirichlet-prod}' (\lambda n. \text{if } n = 1 \text{ then } 1 \text{ else } 0) g x$
using *assms* **by** (*simp add: dirichlet-prod'-assoc dirichlet-prod-commutes*)
also have $\dots = g x$
using *assms* **by** (*subst dirichlet-prod'-one-left*) *auto*
finally show *?thesis ..*
qed

lemma *dirichlet-prod'-inversion*:
assumes $\text{dirichlet-prod } a \text{ainv} = (\lambda n. \text{if } n = 1 \text{ then } 1 \text{ else } 0)$
shows $(\forall x \geq 1. g x = \text{dirichlet-prod}' a f x) \longleftrightarrow (\forall x \geq 1. f x = \text{dirichlet-prod}' \text{ainv } g x)$
using *dirichlet-prod'-inversion1*[*of g a f - ainv*] *dirichlet-prod'-inversion2*[*of f ainv g - a*]
assms **by** *blast*

lemma *dirichlet-prod'-inversion'*:
assumes $a 1 * y = 1$
defines $\text{ainv} \equiv \text{dirichlet-inverse } a y$
shows $(\forall x \geq 1. g x = \text{dirichlet-prod}' a f x) \longleftrightarrow (\forall x \geq 1. f x = \text{dirichlet-prod}' \text{ainv } g x)$
unfolding *ainv-def*
by (*intro dirichlet-prod'-inversion dirichlet-prod-inverse assms*)

lemma *dirichlet-prod'-floor-conv-sum-upto*:
 $\text{dirichlet-prod}' f (\lambda x. \text{real-of-int } (\text{floor } x)) x = \text{sum-upto } (\lambda n. \text{sum-upto } f (x / n)) x$
proof –
have [*simp*]: $\text{sum-upto } (\lambda -. 1 :: \text{real}) x = \text{real } (\text{nat } \lfloor x \rfloor)$ **for** x
by (*simp add: sum-upto-altdef*)
show *?thesis*
using *sum-upto-dirichlet-prod*[*of λn. 1::real f*] *sum-upto-dirichlet-prod*[*of f λn. 1::real*]
by (*simp add: dirichlet-prod'-def dirichlet-prod-commutes*)
qed

lemma (*in completely-multiplicative-function*) *dirichlet-prod-self*:

$\text{dirichlet-prod } f f n = f n * \text{of-nat } (\text{divisor-count } n)$
proof (cases $n = 0$)
 case *False*
 have $\text{dirichlet-prod } f f n = (\sum (r, d) \mid r * d = n. f (r * d))$
 by (simp add: *dirichlet-prod-altdef2 mult*)
 also have $\dots = (\sum (r, d) \mid r * d = n. f n)$
 by (intro *sum.cong*) auto
 also have $\dots = f n * \text{of-nat } (\text{card } \{(r, d). r * d = n\})$
 by (simp add: *mult.commute*)
 also have $\text{bij-betw } \text{fst } \{(r, d). r * d = n\} \{r. r \text{ dvd } n\}$
 by (rule *bij-betwI[of - - - \lambda r. (r, n \text{ div } r)]*) (use *False in auto*)
 hence $\text{card } \{(r, d). r * d = n\} = \text{card } \{r. r \text{ dvd } n\}$
 by (rule *bij-betw-same-card*)
 also have $\dots = \text{divisor-count } n$
 by (simp add: *divisor-count-def*)
 finally show ?thesis .
qed auto

lemma *completely-multiplicative-imp-moebius-mu-inverse:*

fixes $f :: \text{nat} \Rightarrow 'a :: \{\text{comm-ring-1}\}$
 assumes *completely-multiplicative-function* f
 shows $\text{dirichlet-prod } f (\lambda n. \text{moebius-mu } n * f n) n = (\text{if } n = 1 \text{ then } 1 \text{ else } 0)$
proof –
 interpret *completely-multiplicative-function* f **by** *fact*
 have [simp]: $\text{fds } f \neq 0$ **by** (auto simp: *fds-eq-iff*)
 have $\text{dirichlet-prod } f (\lambda n. \text{moebius-mu } n * f n) n =$
 $(\sum (r, d) \mid r * d = n. \text{moebius-mu } r * f (r * d))$
 by (subst *dirichlet-prod-commutes*)
 (simp add: *fds-eq-iff fds-nth-mult fds-nth-fds dirichlet-prod-altdef2 mult-ac*
mult)
 also have $\dots = (\sum (r, d) \mid r * d = n. \text{moebius-mu } r * f n)$
 by (intro *sum.cong*) auto
 also have $\dots = \text{dirichlet-prod } \text{moebius-mu } (\lambda -. 1) n * f n$
 by (simp add: *dirichlet-prod-altdef2 sum-distrib-right case-prod-unfold mult*)
 also have $\text{dirichlet-prod } \text{moebius-mu } (\lambda -. 1) n = \text{fds-nth } (\text{fds } \text{moebius-mu} * \text{fds-zeta}) n$
 by (simp add: *fds-nth-mult*)
 also have $\text{fds } \text{moebius-mu} * \text{fds-zeta} = 1$
 by (simp add: *mult-ac fds-zeta-times-moebius-mu*)
 also have $\text{fds-nth } 1 n * f n = \text{fds-nth } 1 n$
 by (auto simp: *fds-eq-iff fds-nth-one*)
 finally show ?thesis **by** (simp add: *fds-nth-one*)
qed

lemma *dirichlet-prod-inversion-completely-multiplicative:*

fixes $a :: \text{nat} \Rightarrow 'a :: \text{comm-ring-1}$
 assumes *completely-multiplicative-function* a
 shows $(\forall x \geq 1. g x = \text{dirichlet-prod}' a f x) \longleftrightarrow$

$(\forall x \geq 1. f x = \text{dirichlet-prod}' (\lambda n. \text{moebius-mu } n * a n) g x)$

by (*intro dirichlet-prod'-inversion ext completely-multiplicative-imp-moebius-mu-inverse assms*)

lemma *divisor-sigma-conv-dirichlet-prod*:
divisor-sigma $x n = \text{dirichlet-prod} (\lambda n. \text{real } n \text{ powr } x) (\lambda -. 1) n$
proof (*cases n = 0*)
 case *False*
 have *fds* (*divisor-sigma* x) = *fds-shift* x *fds-zeta* * *fds-zeta*
 using *fds-divisor-sigma*[*of x*] **by** (*simp add: mult-ac*)
 thus *?thesis* **using** *False* **by** (*auto simp: fds-eq-iff fds-nth-mult*)
qed *simp-all*

2.2 Legendre's identity

definition *legendre-aux* :: *real* \Rightarrow *nat* \Rightarrow *nat* **where**
legendre-aux $x p = (\text{if prime } p \text{ then } (\sum m \mid m > 0 \wedge \text{real } (p \wedge m) \leq x. \text{nat } \lfloor x / p \wedge m \rfloor) \text{ else } 0)$

lemma *legendre-aux-not-prime* [*simp*]: $\neg \text{prime } p \implies \text{legendre-aux } x p = 0$
by (*simp add: legendre-aux-def*)

lemma *legendre-aux-eq-0*:
 assumes *real* $p > x$
 shows *legendre-aux* $x p = 0$
proof (*cases prime p*)
 case *True*
 have [*simp*]: $\neg \text{real } p \wedge m \leq x \text{ if } m > 0 \text{ for } m$
 proof –
 have $x < \text{real } p \wedge 1$ **using** *assms* **by** *simp*
 also have $\dots \leq \text{real } p \wedge m$
 using *prime-gt-1-nat*[*OF True*] **that** **by** (*intro power-increasing*) *auto*
 finally show *?thesis* **by** *auto*
 qed
 from *assms* **have** $*$: $\{m. m > 0 \wedge \text{real } (p \wedge m) \leq x\} = \{\}$
 using *prime-gt-1-nat*[*OF True*] **by** *auto*
 show *?thesis* **unfolding** *legendre-aux-def*
 by (*subst **) *auto*
qed (*auto simp: legendre-aux-def*)

lemma *legendre-aux-posD*:
 assumes *legendre-aux* $x p > 0$
 shows *prime* p *real* $p \leq x$
proof –
 show *real* $p \leq x$ **using** *legendre-aux-eq-0*[*of x p*] *assms*
 by (*cases real p ≤ x*) *auto*
qed (*use assms in <auto simp: legendre-aux-def split: if-splits>*)

lemma *exponents-le-finite*:

```

assumes  $p > (1 :: \text{nat})$   $k > 0$ 
shows  $\text{finite } \{i. \text{real } (p \wedge (k * i + l)) \leq x\}$ 
proof (rule finite-subset)
show  $\{i. \text{real } (p \wedge (k * i + l)) \leq x\} \subseteq \{.. \text{nat } \lfloor x \rfloor\}$ 
proof safe
  fix  $i$  assume  $i: \text{real } (p \wedge (k * i + l)) \leq x$ 
  have  $i < 2 \wedge i$ 
    by (rule less-exp)
  also from assms have  $i \leq k * i + l$  by (cases  $k$ ) auto
  hence  $2 \wedge i \leq (2 \wedge (k * i + l) :: \text{nat})$ 
    using assms by (intro power-increasing) auto
  also have  $\dots \leq p \wedge (k * i + l)$  using assms by (intro power-mono) auto
  also have  $\text{real } \dots \leq x$  using  $i$  by simp
  finally show  $i \leq \text{nat } \lfloor x \rfloor$  by linarith
qed
qed auto

```

lemma *finite-sum-legendre-aux*:

```

assumes prime  $p$ 
shows  $\text{finite } \{m. m > 0 \wedge \text{real } (p \wedge m) \leq x\}$ 
by (rule finite-subset[OF - exponents-le-finite[where  $k = 1$  and  $l = 0$  and  $p = p$ ]])
  (use assms prime-gt-1-nat[of  $p$ ] in auto)

```

lemma *legendre-aux-set-eq*:

```

assumes prime  $p$   $x \geq 1$ 
shows  $\{m. m > 0 \wedge \text{real } (p \wedge m) \leq x\} = \{0 <.. \text{nat } \lfloor \log (\text{real } p) x \rfloor\}$ 
using prime-gt-1-nat[OF assms(1)] assms
by (auto simp: le-nat-iff le-log-iff le-floor-iff powr-realpow)

```

lemma *legendre-aux-altdef1*:

```

legendre-aux  $x$   $p = (\text{if } \text{prime } p \wedge x \geq 1 \text{ then }
  (\sum_{m \in \{0 <.. \text{nat } \lfloor \log (\text{real } p) x \rfloor\}} \text{nat } \lfloor x / p \wedge m \rfloor) \text{ else } 0)$ 

```

proof (cases $\text{prime } p \wedge x < 1$)

case *False*

thus *?thesis* **using** *legendre-aux-set-eq*[*of* p x] **by** (auto *simp*: *legendre-aux-def*)

next

case *True*

have [*simp*]: $\neg(\text{real } p \wedge m \leq x)$ **for** m

proof –

have $x < \text{real } 1$ **using** *True* **by** *simp*

also have $\text{real } 1 \leq \text{real } (p \wedge m)$

unfolding *of-nat-le-iff* **by** (intro *one-le-power*) (use *prime-gt-1-nat*[*of* p] *True*)

in *auto*)

finally show $\neg(\text{real } p \wedge m \leq x)$ **by** *auto*

qed

have $\{m. m > 0 \wedge \text{real } (p \wedge m) \leq x\} = \{\}$ **by** *simp*

with *True* **show** *?thesis* **by** (*simp* *add*: *legendre-aux-def*)

qed

lemma *legendre-aux-altdef2*:
assumes $x \geq 1$ *prime* p *real* $p \wedge \text{Suc } k > x$
shows $\text{legendre-aux } x \ p = (\sum m \in \{0 <..k\}. \text{nat } \lfloor x / p \wedge m \rfloor)$
proof –
have $\text{legendre-aux } x \ p = (\sum m \mid m > 0 \wedge \text{real } (p \wedge m) \leq x. \text{nat } \lfloor x / p \wedge m \rfloor)$
using *assms* **by** (*simp add: legendre-aux-def*)
also have $\dots = (\sum m \in \{0 <..k\}. \text{nat } \lfloor x / p \wedge m \rfloor)$
proof (*intro sum.mono-neutral-left*)
show $\{m. 0 < m \wedge \text{real } (p \wedge m) \leq x\} \subseteq \{0 <..k\}$
proof *safe*
fix m **assume** $m > 0$ *real* $(p \wedge m) \leq x$
hence $\text{real } p \wedge m \leq x$ **by** *simp*
also note $\langle x < \text{real } p \wedge \text{Suc } k \rangle$
finally show $m \in \{0 <..k\}$ **using** $\langle m > 0 \rangle$
using *prime-gt-1-nat*[*OF* $\langle \text{prime } p \rangle$] **by** (*subst (asm) power-strict-increasing-iff*)
auto
qed
qed (*use prime-gt-0-nat*[*of* p] *assms in* $\langle \text{auto simp: field-simps} \rangle$)
finally show *?thesis* .
qed

theorem *legendre-identity*:
 $\text{sum-upto } \ln x = \text{prime-sum-upto } (\lambda p. \text{legendre-aux } x \ p * \ln p) \ x$
proof –
define S **where** $S = (\text{SIGMA } p:\{p. \text{prime } p \wedge \text{real } p \leq x\}. \{i. i > 0 \wedge \text{real } (p \wedge i) \leq x\})$
have *prime-power-leD*: $\text{real } p \leq x$ **if** $\text{real } p \wedge i \leq x$ *prime* $p \ i > 0$ **for** $p \ i$
proof –
have $\text{real } p \wedge 1 \leq \text{real } p \wedge i$
using *that prime-gt-1-nat*[*of* p] **by** (*intro power-increasing*) *auto*
also have $\dots \leq x$ **by** *fact*
finally show $\text{real } p \leq x$ **by** *simp*
qed
have $\text{sum-upto } \ln x = \text{sum-upto } (\lambda n. \text{mangoldt } n * \text{real } (\text{nat } \lfloor x / \text{real } n \rfloor)) \ x$
by (*rule sum-upto-ln-conv-sum-upto-mangoldt*)
also have $\dots = (\sum (p, i) \mid \text{prime } p \wedge 0 < i \wedge \text{real } (p \wedge i) \leq x. \ln p * \text{real } (\text{nat } \lfloor x / \text{real } (p \wedge i) \rfloor))$
by (*subst sum-upto-primepows*[**where** $g = \lambda p \ i. \ln p * \text{real } (\text{nat } \lfloor x / \text{real } (p \wedge i) \rfloor)$])
(auto simp: mangoldt-non-primepow)
also have $\dots = (\sum (p,i) \in S. \ln p * \text{real } (\text{nat } \lfloor x / p \wedge i \rfloor))$
using *prime-power-leD* **by** (*intro sum.cong refl*) (*auto simp: S-def*)
also have $\dots = (\sum p \mid \text{prime } p \wedge \text{real } p \leq x. \sum i \mid i > 0 \wedge \text{real } (p \wedge i) \leq x. \ln p * \text{real } (\text{nat } \lfloor x / p \wedge i \rfloor))$
proof (*unfold S-def, subst sum.Sigma*)

```

have {p. prime p ∧ real p ≤ x} ⊆ {..nat [x]}
  by (auto simp: le-nat-iff le-floor-iff)
thus finite {p. prime p ∧ real p ≤ x}
  by (rule finite-subset) auto
next
show ∀ p ∈ {p. prime p ∧ real p ≤ x}. finite {i. 0 < i ∧ real (p ^ i) ≤ x}
  by (intro ballI finite-sum-legendre-aux) auto
qed auto
also have ... = (∑ p | prime p ∧ real p ≤ x. ln p *
  real (∑ i | i > 0 ∧ real (p ^ i) ≤ x. (nat [x / p ^ i])))
  by (simp add: sum-distrib-left)
also have ... = (∑ p | prime p ∧ real p ≤ x. ln p * real (legendre-aux x p))
  by (intro sum.cong refl) (auto simp: legendre-aux-def)
also have ... = prime-sum-upto (λp. ln p * real (legendre-aux x p)) x
  by (simp add: prime-sum-upto-def)
finally show ?thesis by (simp add: mult-ac)
qed

lemma legendre-identity':
  fact (nat [x]) = (∏ p | prime p ∧ real p ≤ x. p ^ legendre-aux x p)
proof -
  have fin: finite {p. prime p ∧ real p ≤ x}
    by (rule finite-subset[of - {..nat [x]}]) (auto simp: le-nat-iff le-floor-iff)
  have real (fact (nat [x])) = exp (sum-upto ln x)
    by (subst sum-upto-ln-conv-ln-fact) auto
  also have sum-upto ln x = prime-sum-upto (λp. legendre-aux x p * ln p) x
    by (rule legendre-identity)
  also have exp ... = (∏ p | prime p ∧ real p ≤ x. exp (ln (real p) * legendre-aux
x p))
    unfolding prime-sum-upto-def using fin by (subst exp-sum) (auto simp:
mult-ac)
  also have ... = (∏ p | prime p ∧ real p ≤ x. real (p ^ legendre-aux x p))
proof (intro prod.cong refl)
  fix p assume p ∈ {p. prime p ∧ real p ≤ x}
  hence p > 0 using prime-gt-0-nat[of p] by auto
  from ⟨p > 0⟩ have exp (ln (real p) * legendre-aux x p) = real p powr real
(legendre-aux x p)
    by (simp add: powr-def)
  also from ⟨p > 0⟩ have ... = real (p ^ legendre-aux x p)
    by (subst powr-realpow) auto
  finally show exp (ln (real p) * legendre-aux x p) = ... .
qed
also have ... = real (∏ p | prime p ∧ real p ≤ x. p ^ legendre-aux x p)
  by simp
finally show ?thesis unfolding of-nat-eq-iff .
qed

```

2.3 A weighted sum of the Möbius μ function

context

fixes $M :: \text{real} \Rightarrow \text{real}$

defines $M \equiv (\lambda x. \text{sum-upto } (\lambda n. \text{moebius-mu } n / n) x)$

begin

lemma *abs-sum-upto-moebius-mu-over-n-less*:

assumes $x: x \geq 2$

shows $|M x| < 1$

proof –

have $x * \text{sum-upto } (\lambda n. \text{moebius-mu } n / n) x - \text{sum-upto } (\lambda n. \text{moebius-mu } n * \text{frac } (x / n)) x =$

$\text{sum-upto } (\lambda n. \text{moebius-mu } n * (x / n - \text{frac } (x / n))) x$

by (*subst mult.commute*[of x])

(*simp add: sum-upto-def sum-distrib-right sum-subtractf ring-distrib*)

also have $(\lambda n. x / \text{real } n - \text{frac } (x / \text{real } n)) = (\lambda n. \text{of-int } (\text{floor } (x / \text{real } n)))$

by (*simp add: frac-def*)

also have $\text{sum-upto } (\lambda n. \text{moebius-mu } n * \text{real-of-int } \lfloor x / \text{real } n \rfloor) x =$

$\text{real-of-int } (\text{sum-upto } (\lambda n. \text{moebius-mu } n * \lfloor x / \text{real } n \rfloor) x)$

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

also have $\dots = 1$

using x **by** (*subst sum-upto-moebius-times-floor-linear*) *auto*

finally have $\text{eq}: x * M x = 1 + \text{sum-upto } (\lambda n. \text{moebius-mu } n * \text{frac } (x / n)) x$

by (*simp add: M-def*)

have $x * |M x| = |x * M x|$

using x **by** (*simp add: abs-mult*)

also note *eq*

also have $|1 + \text{sum-upto } (\lambda n. \text{moebius-mu } n * \text{frac } (x / n)) x| \leq$
 $1 + |\text{sum-upto } (\lambda n. \text{moebius-mu } n * \text{frac } (x / n)) x|$

by *linarith*

also have $|\text{sum-upto } (\lambda n. \text{moebius-mu } n * \text{frac } (x / n)) x| \leq$

$\text{sum-upto } (\lambda n. |\text{moebius-mu } n * \text{frac } (x / n)|) x$

unfolding *sum-upto-def* **by** (*rule sum-abs*)

also have $\dots \leq \text{sum-upto } (\lambda n. \text{frac } (x / n)) x$

unfolding *sum-upto-def* **by** (*intro sum-mono*) (*auto simp: moebius-mu-def abs-mult*)

also have $\dots = (\sum n \in \{0 < .. \text{nat } \lfloor x \rfloor\}. \text{frac } (x / n))$

by (*simp add: sum-upto-altdef*)

also have $\{0 < .. \text{nat } \lfloor x \rfloor\} = \text{insert } 1 \{1 < .. \text{nat } \lfloor x \rfloor\}$

using x **by** (*auto simp: le-nat-iff le-floor-iff*)

also have $(\sum n \in \dots \text{frac } (x / n)) = \text{frac } x + (\sum n \in \{1 < .. \text{nat } \lfloor x \rfloor\}. \text{frac } (x / n))$

by (*subst sum.insert*) *auto*

also have $(\sum n \in \{1 < .. \text{nat } \lfloor x \rfloor\}. \text{frac } (x / n)) < (\sum n \in \{1 < .. \text{nat } \lfloor x \rfloor\}. 1)$

using x **by** (*intro sum-strict-mono frac-lt-1*) *auto*

also have $\dots = \text{nat } \lfloor x \rfloor - 1$ **by** *simp*

also have $1 + (\text{frac } x + \text{real } (\text{nat } \lfloor x \rfloor - 1)) = x$

using x **by** (*subst of-nat-diff*) (*auto simp: le-nat-iff le-floor-iff frac-def*)

finally have $x * |M x| < x * 1$ **by** *simp*
with x **show** $|M x| < 1$
by (*subst (asm) mult-less-cancel-left-pos*) *auto*
qed

lemma *sum-upto-moebius-mu-over-n-eq*:
assumes $x < 2$
shows $M x = (\text{if } x \geq 1 \text{ then } 1 \text{ else } 0)$
proof (*cases x ≥ 1*)
case *True*
have $M x = (\sum n \in \{n. n > 0 \wedge \text{real } n \leq x\}. \text{moebius-mu } n / n)$
by (*simp add: M-def sum-upto-def*)
also from *assms True* **have** $\{n. n > 0 \wedge \text{real } n \leq x\} = \{1\}$
by *auto*
thus *?thesis* **using** *True* **by** (*simp add: M-def sum-upto-def*)
next
case *False*
have $M x = (\sum n \in \{n. n > 0 \wedge \text{real } n \leq x\}. \text{moebius-mu } n / n)$
by (*simp add: M-def sum-upto-def*)
also from *False* **have** $\{n. n > 0 \wedge \text{real } n \leq x\} = \{\}$
by *auto*
finally show *?thesis* **using** *False* **by** (*simp add: M-def*)
qed

lemma *abs-sum-upto-moebius-mu-over-n-le*: $|M x| \leq 1$
using *sum-upto-moebius-mu-over-n-eq[of x]* *abs-sum-upto-moebius-mu-over-n-less[of x]*
by (*cases x < 2*) *auto*

end

end

3 The Prime ω function

theory *Primes-Omega*
imports *Dirichlet-Series.Dirichlet-Series Dirichlet-Series.Divisor-Count*
begin

The prime ω function $\omega(n)$ counts the number of distinct prime factors of n .

definition *primes-omega* :: $\text{nat} \Rightarrow \text{nat}$ **where**
primes-omega $n = \text{card } (\text{prime-factors } n)$

lemma *primes-omega-prime* [*simp*]: $\text{prime } p \implies \text{primes-omega } p = 1$
by (*simp add: primes-omega-def prime-factorization-prime*)

lemma *primes-omega-0* [*simp*]: $\text{primes-omega } 0 = 0$
by (*simp add: primes-omega-def*)

lemma *primes-omega-1* [*simp*]: $\text{primes-omega } 1 = 0$
by (*simp add: primes-omega-def*)

lemma *primes-omega-Suc-0* [*simp*]: $\text{primes-omega } (\text{Suc } 0) = 0$
by (*simp add: primes-omega-def*)

lemma *primes-omega-power* [*simp*]: $n > 0 \implies \text{primes-omega } (x \wedge n) = \text{primes-omega } x$
by (*simp add: primes-omega-def prime-factors-power*)

lemma *primes-omega-primelow* [*simp*]: $\text{primelow } n \implies \text{primes-omega } n = 1$
by (*auto simp: primelow-def*)

lemma *primes-omega-eq-0-iff*: $\text{primes-omega } n = 0 \iff n = 0 \vee n = 1$
by (*auto simp: primes-omega-def prime-factorization-empty-iff*)

lemma *primes-omega-pos* [*simp, intro*]: $n > 1 \implies \text{primes-omega } n > 0$
by (*cases primes-omega n > 0*) (*auto simp: primes-omega-eq-0-iff*)

lemma *primes-omega-mult-coprime*:
assumes *coprime x y x > 0 \vee y > 0*
shows $\text{primes-omega } (x * y) = \text{primes-omega } x + \text{primes-omega } y$
proof (*cases x = 0 \vee y = 0*)
case *False*
hence $\text{prime-factors } (x * y) = \text{prime-factors } x \cup \text{prime-factors } y$
by (*subst prime-factorization-mult*) *auto*
also {
have $\text{prime-factors } x \cap \text{prime-factors } y = \text{set-mset } (\text{prime-factorization } (\text{gcd } x y))$
using *False* **by** (*subst prime-factorization-gcd*) *auto*
also have $\text{gcd } x y = 1$ **using** *coprime x y* **by** *auto*
finally have $\text{card } (\text{prime-factors } x \cup \text{prime-factors } y) = \text{primes-omega } x + \text{primes-omega } y$
unfolding *primes-omega-def* **by** (*intro card-Un-disjoint*) (*use False in auto*)
}
finally show *?thesis* **by** (*simp add: primes-omega-def*)
qed (*use assms in auto*)

lemma *divisor-count-squarefree*:
assumes *squarefree n n > 0*
shows $\text{divisor-count } n = 2 \wedge \text{primes-omega } n$
proof –
have $\text{divisor-count } n = (\prod_{p \in \text{prime-factors } n} \text{Suc } (\text{multiplicity } p n))$
using *assms* **by** (*subst divisor-count.prod-prime-factors'*) *auto*
also have $\dots = (\prod_{p \in \text{prime-factors } n} 2)$
using *assms* *assms* **by** (*intro prod.cong*) (*auto simp: squarefree-factorial-semiring'*)
finally show *?thesis* **by** (*simp add: primes-omega-def*)
qed

end

4 The Primorial function

theory *Primorial*
 imports *Prime-Distribution-Elementary-Library Primes-Omega*
begin

4.1 Definition and basic properties

definition *primorial* :: *real* \Rightarrow *nat* **where**
 primorial $x = \prod \{p. \text{prime } p \wedge \text{real } p \leq x\}$

lemma *primorial-mono*: $x \leq y \implies \text{primorial } x \leq \text{primorial } y$
 unfolding *primorial-def*
 by (*intro dvd-imp-le prod-dvd-prod-subset*)
 (*auto intro!: prod-pos finite-primes-le dest: prime-gt-0-nat*)

lemma *prime-factorization-primorial*:
 prime-factorization (*primorial* x) = *mset-set* $\{p. \text{prime } p \wedge \text{real } p \leq x\}$
proof (*intro multiset-eqI*)
 fix $p :: \text{nat}$
 note $\text{fin} = \text{finite-primes-le}[of\ x]$
 show $\text{count} (\text{prime-factorization} (\text{primorial } x))\ p =$
 $\text{count} (\text{mset-set } \{p. \text{prime } p \wedge \text{real } p \leq x\})\ p$
 proof (*cases prime p*)
 case *True*
 hence $\text{count} (\text{prime-factorization} (\text{primorial } x))\ p =$
 $\text{sum} (\text{multiplicity } p)\ \{p. \text{prime } p \wedge \text{real } p \leq x\}$
 unfolding *primorial-def count-prime-factorization* **using** *fin*
 by (*subst prime-elem-multiplicity-prod-distrib*) *auto*
 also from *True* **have** $\dots = \text{sum } (\lambda-. 1)\ (\text{if } p \leq x \text{ then } \{p\} \text{ else } \{\})$ **using** *fin*
 by (*intro sum.mono-neutral-cong-right*) (*auto simp: prime-multiplicity-other*
split: if-splits)
 also have $\dots = \text{count} (\text{mset-set } \{p. \text{prime } p \wedge \text{real } p \leq x\})\ p$
 using *True fin* **by** *auto*
 finally show *?thesis* .
 qed *auto*
qed

lemma *prime-factors-primorial* [*simp*]:
 prime-factors (*primorial* x) = $\{p. \text{prime } p \wedge \text{real } p \leq x\}$
 unfolding *prime-factorization-primorial* **using** *finite-primes-le[of x]* **by** *simp*

lemma *primorial-pos* [*simp, intro*]: *primorial* $x > 0$
 unfolding *primorial-def* **by** (*intro prod-pos*) (*auto dest: prime-gt-0-nat*)

lemma *primorial-neq-zero* [*simp*]: *primorial* $x \neq 0$

by *auto*

lemma *of-nat-primes-omega-primorial* [*simp*]: $\text{real } (\text{primes-omega } (\text{primorial } x)) = \text{primes-pi } x$
 by (*simp add: primes-omega-def primes-pi-def prime-sum-upto-def*)

lemma *primes-omega-primorial*: $\text{primes-omega } (\text{primorial } x) = \text{nat } \lfloor \text{primes-pi } x \rfloor$
 by (*simp add: primes-omega-def primes-pi-def prime-sum-upto-def*)

lemma *prime-dvd-primorial-iff*: $\text{prime } p \implies p \text{ dvd primorial } x \iff p \leq x$
 using *finite-primes-le*[*of x*]
 by (*auto simp: primorial-def prime-dvd-prod-iff dest: primes-dvd-imp-eq*)

lemma *squarefree-primorial* [*intro*]: *squarefree* (*primorial x*)
 unfolding *primorial-def*
 by (*intro squarefree-prod-coprime*) (*auto simp: squarefree-prime intro: primes-coprime*)

lemma *primorial-ge*: $\text{primorial } x \geq 2 \text{ powr primes-pi } x$
proof –
 have $2 \text{ powr primes-pi } x = \text{real } (\prod p \mid \text{prime } p \wedge \text{real } p \leq x. 2)$
 by (*simp add: primes-pi-def prime-sum-upto-def powr-realpow*)
 also have $(\prod p \mid \text{prime } p \wedge \text{real } p \leq x. 2) \leq (\prod p \mid \text{prime } p \wedge \text{real } p \leq x. p)$
 by (*intro prod-mono*) (*auto dest: prime-gt-1-nat*)
 also have $\dots = \text{primorial } x$ by (*simp add: primorial-def*)
 finally show ?thesis by *simp*

qed

lemma *primorial-at-top*: *filterlim primorial at-top at-top*
proof –
 have *filterlim* ($\lambda x. \text{real } (\text{primorial } x)$) *at-top at-top*
proof (*rule filterlim-at-top-mono*)
 show *eventually* ($\lambda x. \text{primorial } x \geq 2 \text{ powr primes-pi } x$) *at-top*
 by (*intro always-eventually primorial-ge allI*)
 have *filterlim* ($\lambda x. \text{exp } (\ln 2 * \text{primes-pi } x)$) *at-top at-top*
 by (*intro filterlim-compose*[*OF exp-at-top*]
filterlim-tendsto-pos-mult-at-top[*OF tendsto-const*] π -*at-top*) *auto*
 thus *filterlim* ($\lambda x. 2 \text{ powr primes-pi } x$) *at-top at-top*
 by (*simp add: powr-def mult-ac*)

qed

thus ?thesis unfolding *filterlim-sequentially-iff-filterlim-real* [*symmetric*].

qed

lemma *totient-primorial*:
 $\text{real } (\text{totient } (\text{primorial } x)) =$
 $\text{real } (\text{primorial } x) * (\prod p \mid \text{prime } p \wedge \text{real } p \leq x. 1 - 1 / \text{real } p)$ **for** x
proof –
 have $\text{real } (\text{totient } (\text{primorial } x)) =$
 $\text{primorial } x * (\prod p \in \text{prime-factors } (\text{primorial } x). 1 - 1 / p)$
 by (*rule totient-formula2*)

thus *?thesis* by *simp*
qed

lemma *ln-primorial*: $\ln (\text{primorial } x) = \text{primes-theta } x$

proof –

have *finite* $\{p. \text{prime } p \wedge \text{real } p \leq x\}$

by (*rule finite-subset*[of - $\{..nat \lfloor x \rfloor\}$]) (*auto simp: le-nat-iff le-floor-iff*)

thus *?thesis* **unfolding** *of-nat-prod primorial-def*

by (*subst ln-prod*) (*auto dest: prime-gt-0-nat simp: v-def prime-sum-upto-def*)

qed

lemma *divisor-count-primorial*: $\text{divisor-count } (\text{primorial } x) = 2^{\text{powr primes-pi } x}$

proof –

have $\text{divisor-count } (\text{primorial } x) = (\prod p \mid \text{prime } p \wedge \text{real } p \leq x. \text{divisor-count } p)$

unfolding *primorial-def*

by (*subst divisor-count.prod-coprime*) (*auto simp: primes-coprime*)

also have $\dots = (\prod p \mid \text{prime } p \wedge \text{real } p \leq x. 2)$

by (*intro prod.cong divisor-count.prime*) *auto*

also have $\dots = 2^{\text{powr primes-pi } x}$

by (*simp add: primes-pi-def prime-sum-upto-def powr-realpow*)

finally show *?thesis* .

qed

4.2 An alternative view on primorials

The following function is an alternative representation of primorials; instead of taking the product of all primes up to a given real bound x , it takes the product of the first k primes. This is sometimes more convenient.

definition *primorial'* :: $\text{nat} \Rightarrow \text{nat}$ **where**

primorial' n = ($\prod k < n. \text{nth-prime } k$)

lemma *primorial'-0* [*simp*]: *primorial' 0 = 1*

and *primorial'-1* [*simp*]: *primorial' 1 = 2*

and *primorial'-2* [*simp*]: *primorial' 2 = 6*

and *primorial'-3* [*simp*]: *primorial' 3 = 30*

by (*simp-all add: primorial'-def lessThan-nat-numeral*)

lemma *primorial'-Suc*: $\text{primorial}' (\text{Suc } n) = \text{nth-prime } n * \text{primorial}' n$

by (*simp add: primorial'-def*)

lemma *primorial'-pos* [*intro*]: *primorial' n > 0*

unfolding *primorial'-def* **by** (*auto intro: prime-gt-0-nat*)

lemma *primorial'-neq-0* [*simp*]: *primorial' n \neq 0*

by *auto*

lemma *strict-mono-primorial'*: *strict-mono primorial'*

unfolding *strict-mono-Suc-iff*

proof

```

fix n :: nat
have primorial' n * 1 < primorial' n * nth-prime n
  using prime-gt-1-nat[OF prime-nth-prime[of n]]
  by (intro mult-strict-left-mono) auto
thus primorial' n < primorial' (Suc n)
  by (simp add: primorial'-Suc)
qed

lemma prime-factorization-primorial':
  prime-factorization (primorial' k) = mset-set (nth-prime ' {..<k})
proof -
  have prime-factorization (primorial' k) = (∑ i<k. prime-factorization (nth-prime
  i))
  unfolding primorial'-def by (subst prime-factorization-prod) (auto intro: prime-gt-0-nat)
  also have ... = (∑ i<k. {#nth-prime i#})
  by (intro sum.cong prime-factorization-prime) auto
  also have ... = (∑ p∈nth-prime ' {..<k}. {#p#})
  by (subst sum.reindex) (auto intro: inj-onI)
  also have ... = mset-set (nth-prime ' {..<k})
  by simp
  finally show ?thesis .
qed

lemma prime-factors-primorial': prime-factors (primorial' k) = nth-prime ' {..<k}
  by (simp add: prime-factorization-primorial')

lemma primes-omega-primorial' [simp]: primes-omega (primorial' k) = k
  unfolding primes-omega-def prime-factors-primorial' by (subst card-image) (auto
  intro: inj-onI)

lemma squarefree-primorial' [intro]: squarefree (primorial' x)
  unfolding primorial'-def
  by (intro squarefree-prod-coprime) (auto intro: squarefree-prime intro: primes-coprime)

lemma divisor-count-primorial' [simp]: divisor-count (primorial' k) = 2 ^ k
  by (subst divisor-count-squarefree) auto

lemma totient-primorial':
  totient (primorial' k) = primorial' k * (∏ i<k. 1 - 1 / nth-prime i)
  unfolding totient-formula2 prime-factors-primorial'
  by (subst prod.reindex) (auto intro: inj-onI)

lemma primorial-conv-primorial': primorial x = primorial' (nat [primes-pi x])
  unfolding primorial-def primorial'-def
proof (rule prod.reindex-bij-witness[of - nth-prime λp. nat [primes-pi (real p)] -
  1])
  fix p assume p: p ∈ {p. prime p ∧ real p ≤ x}
  have [simp]: primes-pi 2 = 1 by (auto simp: eval-π)
  have primes-pi p ≥ 1

```

```

    using p  $\pi$ -mono[of 2 real p] by (auto dest!: prime-gt-1-nat)
  with p show nth-prime (nat [primes-pi p] - 1) = p
    using  $\pi$ -pos[of real p]
    by (intro nth-prime-eqI'')
      (auto simp: le-nat-iff le-floor-iff primes-pi-def prime-sum-upto-def of-nat-diff)
  from p have nat [primes-pi (real p)]  $\leq$  nat [primes-pi x]
    by (intro nat-mono floor-mono  $\pi$ -mono) auto
  hence nat [primes-pi (real p)] - 1 < nat [primes-pi x]
    using  $\langle$ primes-pi p  $\geq$  1 $\rangle$  by linarith
  thus nat [primes-pi (real p)] - 1  $\in$  {.. $\leq$ nat [primes-pi x]} by simp
  show nth-prime (nat [primes-pi (real p)] - 1) = p
    using p  $\langle$ primes-pi p  $\geq$  1 $\rangle$ 
    by (intro nth-prime-eqI'') (auto simp: le-nat-iff primes-pi-def prime-sum-upto-def)
next
fix k assume k: k  $\in$  {.. $\leq$ nat [primes-pi x]}
thus *: nat [primes-pi (real (nth-prime k))] - 1 = k by auto
from k have  $\neg$ (x < 2) by (intro notI) auto
hence x  $\geq$  2 by simp
have real (nth-prime k)  $\leq$  real (nth-prime (nat [primes-pi x] - 1))
  using k by simp
also have ...  $\leq$  x
  using nth-prime-partition''[of x]  $\langle$ x  $\geq$  2 $\rangle$  by auto
finally show nth-prime k  $\in$  {p. prime p  $\wedge$  real p  $\leq$  x}
  by auto
qed

```

lemma primorial'-conv-primorial:

```

  assumes n > 0
  shows primorial' n = primorial (nth-prime (n - 1))
proof -
  have primorial (nth-prime (n - 1)) = ( $\prod$  k<nat (int (n - 1) + 1). nth-prime
k)
    by (simp add: primorial-conv-primorial' primorial'-def)
  also have nat (int (n - 1) + 1) = n
    using assms by auto
  finally show ?thesis by (simp add: primorial'-def)
qed

```

4.3 Maximal compositeness of primorials

Primorials are maximally composite, i. e. any number with k distinct prime factors is as least as big as the primorial with k distinct prime factors, and any number less than a primorial has strictly less prime factors.

lemma nth-prime-le-prime-sequence:

```

  fixes p :: nat  $\Rightarrow$  nat
  assumes strict-mono-on p {.. $\leq$ n} and  $\bigwedge$ k. k < n  $\implies$  prime (p k) and k < n
  shows nth-prime k  $\leq$  p k
  using assms(3)
proof (induction k)

```

```

case 0
hence prime (p 0) by (intro assms)
hence  $p \geq 2$  by (auto dest: prime-gt-1-nat)
thus ?case by simp
next
case (Suc k)
have IH:  $Suc (nth\text{-prime } k) \leq Suc (p k)$  using Suc by simp
have  $nth\text{-prime } (Suc k) = smallest\text{-prime-beyond } (Suc (nth\text{-prime } k))$ 
  by (simp add: nth-prime-Suc)
also {
  have  $Suc (nth\text{-prime } k) \leq Suc (p k)$  using Suc by simp
  also have  $\dots \leq smallest\text{-prime-beyond } (Suc (p k))$ 
    by (rule smallest-prime-beyond-le)
  finally have  $smallest\text{-prime-beyond } (Suc (nth\text{-prime } k)) \leq smallest\text{-prime-beyond } (Suc (p k))$ 
    by (rule smallest-prime-beyond-smallest[OF prime-smallest-prime-beyond])
}
also have  $p k < p (Suc k)$ 
  using Suc by (intro strict-mono-onD[OF assms(1)]) auto
hence  $smallest\text{-prime-beyond } (Suc (p k)) \leq p (Suc k)$ 
  using Suc.prem by (intro smallest-prime-beyond-smallest assms) auto
finally show ?case .
qed

```

theorem *primorial'-primes-omega-le:*

```

fixes n :: nat
assumes n:  $n > 0$ 
shows primorial' (primes-omega n)  $\leq n$ 
proof (cases n = 1)
case True
thus ?thesis by simp
next
case False
with assms have  $n > 1$  by simp
define m where  $m = primes\text{-omega } n$ 
define P where  $P = \{p. prime\ p \wedge real\ p \leq primes\text{-pi } n\}$ 
define ps where  $ps = sorted\text{-list-of-set } (prime\text{-factors } n)$ 
have set-ps:  $set\ ps = prime\text{-factors } n$  by (simp add: ps-def)
have [simp]:  $length\ ps = m$  by (simp add: ps-def m-def primes-omega-def)
have sorted ps distinct ps by (simp-all add: ps-def)
hence mono: strict-mono-on  $(\lambda k. ps ! k) \{..<m\}$ 
  by (intro strict-mono-onI le-neq-trans) (auto simp: sorted-nth-mono distinct-conv-nth)
from  $\langle n > 1 \rangle$  have  $m > 0$ 
  by (auto simp: m-def prime-factorization-empty-iff intro!: Nat.gr0I)

have primorial' m =  $(\prod_{k < m. nth\text{-prime } k})$ 
  using  $\langle m > 0 \rangle$  by (simp add: of-nat-diff primorial'-def m-def)
also have  $(\prod_{k < m. nth\text{-prime } k) \leq (\prod_{k < m. ps ! k} multiplicity (ps ! k) n)$ 
  proof (intro prod-mono conjI)

```



```

fix  $i$  assume  $i: i \in \{..<m\}$ 
hence  $p: ps ! i \in \text{prime-factors } n$ 
using  $\text{set-ps}$  by  $(\text{auto simp: set-conv-nth})$ 
with  $i \text{ set-ps}$  have  $\text{nth-prime } i \leq ps ! i$ 
by  $(\text{intro nth-prime-le-prime-sequence}[\text{where } n = m] \text{ mono}) (\text{auto simp:}$ 
 $\text{set-conv-nth})$ 
also have  $\dots \leq ps ! i \wedge \text{multiplicity } (ps ! i) n$ 
using  $p$  by  $(\text{intro self-le-power}) (\text{auto simp: prime-factors-multiplicity dest:}$ 
 $\text{prime-gt-1-nat})$ 
finally show  $\text{nth-prime } i \leq \dots$  .
qed auto
also have  $\dots = (\prod_{p \in (\lambda i. ps ! i) \text{ ' } \{..<m\}. p \wedge \text{multiplicity } p n)$ 
using  $\langle \text{distinct } ps \rangle$  by  $(\text{subst prod.reindex}) (\text{auto intro!: inj-onI simp: dis-}$ 
 $\text{tinct-conv-nth})$ 
also have  $(\lambda i. ps ! i) \text{ ' } \{..<m\} = \text{set } ps$ 
by  $(\text{auto simp: set-conv-nth})$ 
also have  $\text{set } ps = \text{prime-factors } n$ 
by  $(\text{simp add: set-ps})$ 
also have  $(\prod_{p \in \text{prime-factors } n. p \wedge \text{multiplicity } p n) = n$ 
using  $\langle n > 1 \rangle$  by  $(\text{intro prime-factorization-nat } [\text{symmetric}]) \text{ auto}$ 
finally show  $\text{primorial}' m \leq n$  .
qed

```

lemma *primes-omega-less-primes-omega-primorial*:

```

fixes  $n :: \text{nat}$ 
assumes  $n: n > 0$  and  $n < \text{primorial } x$ 
shows  $\text{primes-omega } n < \text{primes-omega } (\text{primorial } x)$ 
proof  $(\text{cases } n > 1)$ 
case False
have  $[\text{simp}]: \text{primes-pi } 2 = 1$  by  $(\text{simp add: eval-}\pi)$ 
from False assms have  $[\text{simp}]: n = 1$  by auto
from assms have  $\neg(x < 2)$ 
by  $(\text{intro notI}) (\text{auto simp: primorial-conv-primorial}' )$ 
thus ?thesis using assms  $\pi$ -mono $[\text{of } 2 \ x]$  by auto
next
case True
define  $m$  where  $m = \text{primes-omega } n$ 
have  $le: \text{primorial}' m \leq n$ 
using  $\text{primorial}'\text{-primes-omega-le}[\text{of } n] \langle n > 1 \rangle$  by  $(\text{simp add: } m\text{-def } \text{primes-omega-def})$ 
also have  $\dots < \text{primorial } x$  by fact
also have  $\dots = \text{primorial}' (\text{nat } \lfloor \text{primes-pi } x \rfloor)$ 
by  $(\text{simp add: primorial-conv-primorial}' )$ 
finally have  $m < \text{nat } \lfloor \text{primes-pi } x \rfloor$ 
using strict-mono-less $[\text{OF } \text{strict-mono-primorial}' ]$  by simp
hence  $m < \text{primes-pi } x$  by linarith
also have  $\dots = \text{primes-omega } (\text{primorial } x)$  by simp
finally show ?thesis unfolding  $m\text{-def}$  of-nat-less-iff .
qed

```

```

lemma primes-omega-le-primes-omega-primorial:
  fixes  $n :: \text{nat}$ 
  assumes  $n \leq \text{primorial } x$ 
  shows  $\text{primes-omega } n \leq \text{primes-omega } (\text{primorial } x)$ 
proof –
  consider  $n = 0 \mid n > 0 \mid n = \text{primorial } x \mid n > 0 \mid n \neq \text{primorial } x$  by force
  thus ?thesis
  by cases (use primes-omega-less-primes-omega-primorial[of  $n \ x$ ] assms in auto)
qed

end

```

5 The LCM of the first n natural numbers

```

theory Lcm-Nat-Upto
  imports Prime-Number-Theorem.Prime-Counting-Functions
begin

```

In this section, we examine $\text{Lcm } \{1..n\}$. In particular, we will show that it is equal to $e^{\psi(n)}$ and thus (by the PNT) $e^{n+o(n)}$.

```

lemma multiplicity-Lcm:
  fixes  $A :: 'a :: \{\text{semiring-Gcd, factorial-semiring-gcd}\}$  set
  assumes  $\text{finite } A \ A \neq \{\}$   $\text{prime } p \ 0 \notin A$ 
  shows  $\text{multiplicity } p \ (\text{Lcm } A) = \text{Max } (\text{multiplicity } p \ ` A)$ 
  using assms
proof (induction A rule: finite-ne-induct)
  case (insert x A)
  have  $\text{Lcm } (\text{insert } x \ A) = \text{lcm } x \ (\text{Lcm } A)$  by simp
  also have  $\text{multiplicity } p \ \dots = \text{Max } (\text{multiplicity } p \ ` \text{insert } x \ A)$ 
    using insert by (subst multiplicity-lcm) (auto simp: Lcm-0-iff)
  finally show ?case by simp
qed auto

```

The multiplicity of any prime p in $\text{Lcm } \{1..n\}$ differs from that in $\text{Lcm } \{1..n-1\}$ iff n is a power of p , in which case it is greater by 1.

```

lemma multiplicity-Lcm-atLeast1AtMost-Suc:
  fixes  $p \ n :: \text{nat}$ 
  assumes  $p$ : prime  $p$  and  $n$ :  $n > 0$ 
  shows  $\text{multiplicity } p \ (\text{Lcm } \{1..\text{Suc } n\}) =$ 
    (if  $\exists k. \text{Suc } n = p \wedge k$  then 1 else 0) +  $\text{multiplicity } p \ (\text{Lcm } \{1..n\})$ 
proof –
  define  $k$  where  $k = \text{Max } (\text{multiplicity } p \ ` \{1..n\})$ 
  define  $l$  where  $l = \text{multiplicity } p \ (\text{Suc } n)$ 
  have eq:  $\{1..\text{Suc } n\} = \text{insert } (\text{Suc } n) \ \{1..n\}$  by auto
  from  $\langle \text{prime } p \rangle$  have  $p > 1$  by (auto dest: prime-gt-1-nat)

  have  $\text{multiplicity } p \ (\text{Lcm } \{1..\text{Suc } n\}) = \text{Max } (\text{multiplicity } p \ ` \{1..\text{Suc } n\})$ 
    using assms by (subst multiplicity-Lcm) auto

```

also have $\text{multiplicity } p \text{ ' } \{1..Suc\ n\} =$
 $\text{insert } (\text{multiplicity } p \text{ (Suc } n)) \text{ (multiplicity } p \text{ ' } \{1..n\})$
by *(simp only: eq image-insert)*
also have $\text{Max } \dots = \max\ l\ k$
unfolding *l-def k-def* **using** *assms* **by** *(subst Max.insert) auto*
also have $\dots = (\text{if } \exists k. \text{Suc } n = p \wedge k \text{ then } 1 \text{ else } 0) + k$
proof *(cases $\exists k. \text{Suc } n = p \wedge k$)*
case *False*
have $p \wedge l \text{ dvd } \text{Suc } n$
unfolding *l-def* **by** *(intro multiplicity-dvd)*
hence $p \wedge l \leq \text{Suc } n$
unfolding *l-def* **by** *(intro dvd-imp-le multiplicity-dvd) auto*
moreover have $\text{Suc } n \neq p \wedge l$ **using** *False* **by** *blast*
ultimately have $p \wedge l < \text{Suc } n$ **by** *linarith*
moreover have $p \wedge l > 0$ **using** $\langle p > 1 \rangle$ **by** *(intro zero-less-power) auto*
ultimately have $l = \text{multiplicity } p \text{ (} p \wedge l \text{)}$ **and** $p \wedge l \in \{1..n\}$
using $\langle \text{prime } p \rangle$ **by** *auto*
hence $l \leq k$ **unfolding** *k-def* **by** *(intro Max.coboundedI) auto*
with *False* **show** *?thesis* **by** *(simp add: l-def k-def)*
next
case *True*
then obtain x **where** $\text{Suc } n = p \wedge x$ **by** *blast*
from x **and** $\langle n > 0 \rangle$ **have** $x > 0$ **by** *(intro Nat.gr0I) auto*
from x **have** $[simp]: l = x$ **using** $\langle \text{prime } p \rangle$ **by** *(simp add: l-def)*
have $x = k + 1$
proof *(intro antisym)*
have $p \wedge (x - 1) < \text{Suc } n$
using $\langle x > 0 \rangle \langle p > 1 \rangle$ **unfolding** x **by** *(intro power-strict-increasing) auto*
moreover have $p \wedge (x - 1) > 0$
using $\langle p > 1 \rangle$ **by** *(intro zero-less-power) auto*
ultimately have $\text{multiplicity } p \text{ (} p \wedge (x - 1) \text{)} = x - 1$ **and** $p \wedge (x - 1) \in$
 $\{1..n\}$
using $\langle \text{prime } p \rangle$ **by** *auto*
hence $x - 1 \leq k$
unfolding *k-def* **by** *(intro Max.coboundedI) force+*
thus $x \leq k + 1$ **by** *linarith*
next
have $\text{multiplicity } p\ y < x$ **if** $y \in \{1..n\}$ **for** y
proof $-$
have $p \wedge \text{multiplicity } p\ y \leq y$
using *that* **by** *(intro dvd-imp-le multiplicity-dvd) auto*
also have $\dots < \text{Suc } n$ **using** *that* **by** *simp*
also have $\dots = p \wedge x$ **by** *(fact x)*
finally show $\text{multiplicity } p\ y < x$
using $\langle p > 1 \rangle$ **by** *(subst (asm) power-strict-increasing-iff)*
qed
hence $k < x$
using $\langle n > 0 \rangle$ **unfolding** *k-def* **by** *(subst Max-less-iff) auto*
thus $k + 1 \leq x$ **by** *simp*

```

    qed
    thus ?thesis using True by simp
  qed
  also have  $k = \text{multiplicity } p \text{ (Lcm } \{1..n\})$ 
    unfolding  $k\text{-def}$  using  $\langle n > 0 \rangle$  and  $\langle \text{prime } p \rangle$  by (subst multiplicity-Lcm)
  auto
  finally show ?thesis .
  qed

```

Consequently, $\text{Lcm } \{1..n\}$ differs from $\text{Lcm } \{1..n - 1\}$ iff n is of the form p^k for some prime p , in which case it is greater by a factor of p .

lemma *Lcm-atLeast1AtMost-Suc*:

$\text{Lcm } \{1..\text{Suc } n\} = \text{Lcm } \{1..n\} * (\text{if primepow } (\text{Suc } n) \text{ then aprime divisor } (\text{Suc } n) \text{ else } 1)$

proof (cases $n > 0$)

case True

show ?thesis

proof (rule multiplicity-eq-nat)

fix $p :: \text{nat}$ assume prime p

define x where $x = (\text{if primepow } (\text{Suc } n) \text{ then aprime divisor } (\text{Suc } n) \text{ else } 1)$

have $x > 0$

using $\langle n > 0 \rangle$ by (auto simp: $x\text{-def}$ intro!: aprime divisor-pos-nat)

have $\text{multiplicity } p \text{ (Lcm } \{1..n\} * x) = \text{multiplicity } p \text{ (Lcm } \{1..n\}) + \text{multiplicity } p x$

using $\langle \text{prime } p \rangle \langle x > 0 \rangle$ by (subst prime-elem-multiplicity-mult-distrib) auto

also consider $\exists k. \text{Suc } n = p ^ k \mid \text{primepow } (\text{Suc } n) \neg(\exists k. \text{Suc } n = p ^ k) \mid \neg\text{primepow } (\text{Suc } n)$ by blast

hence $\text{multiplicity } p x = (\text{if } \exists k. \text{Suc } n = p ^ k \text{ then } 1 \text{ else } 0)$

proof cases

assume $\exists k. \text{Suc } n = p ^ k$

thus ?thesis using $\langle \text{prime } p \rangle \langle n > 0 \rangle$

by (auto simp: $x\text{-def}$ aprime divisor-prime-power intro!: Nat.gr0I)

next

assume *: $\text{primepow } (\text{Suc } n) \nexists k. \text{Suc } n = p ^ k$

then obtain $q k$ where $qk: \text{prime } q \text{ Suc } n = q ^ k k > 0 q \neq p$

by (auto simp: primepow-def)

thus ?thesis using $\langle \text{prime } p \rangle$

by (subst *) (auto simp: $x\text{-def}$ aprime divisor-prime-power prime-multiplicity-other)

next

assume *: $\neg\text{primepow } (\text{Suc } n)$

hence **: $\nexists k. \text{Suc } n = p ^ k$ using $\langle \text{prime } p \rangle \langle n > 0 \rangle$ by auto

from * show ?thesis unfolding $x\text{-def}$

by (subst **) auto

qed

also have $\text{multiplicity } p \text{ (Lcm } \{1..n\}) + \dots = \text{multiplicity } p \text{ (Lcm } \{1..\text{Suc } n\})$

using $\langle \text{prime } p \rangle \langle n > 0 \rangle$ by (subst multiplicity-Lcm-atLeast1AtMost-Suc)

(auto simp: $x\text{-def}$)

finally show $\text{multiplicity } p \text{ (Lcm } \{1..\text{Suc } n\}) = \text{multiplicity } p \text{ (Lcm } \{1..n\}) *$

```

x) ..
  qed (use ⟨n > 0⟩ in ⟨auto intro!: Nat.gr0I dest: aprime divisor-pos-nat⟩)
qed auto

```

It follows by induction that $\text{Lcm } \{1..n\} = e^{\psi(n)}$.

```

lemma Lcm-atLeast1AtMost-conv-ψ:
  includes prime-counting-notation
  shows real (Lcm {1..n}) = exp (ψ (real n))
proof (induction n)
  case (Suc n)
  have real (Lcm {1..Suc n}) =
    real (Lcm {1..n}) * (if primepow (Suc n) then aprime divisor (Suc n) else
1)
  by (subst Lcm-atLeast1AtMost-Suc) auto
  also {
    assume primepow (Suc n)
    hence Suc n > Suc 0 by (rule primepow-gt-Suc-0)
    hence aprime divisor (Suc n) > 0 by (intro aprime divisor-pos-nat) auto
  }
  hence (if primepow (Suc n) then aprime divisor (Suc n) else 1) = exp (mangoldt
(Suc n))
  by (simp add: mangoldt-def)
  also have Lcm {1..n} * ... = exp (ψ (real n + 1))
  using Suc.IH by (simp add: primes-psi-def sum-upto-plus1 exp-add)
  finally show ?case by (simp add: add-ac)
qed simp-all

```

```

lemma Lcm-upto-real-conv-ψ:
  includes prime-counting-notation
  shows real (Lcm {1..nat ⌊x⌋}) = exp (ψ x)
  by (subst Lcm-atLeast1AtMost-conv-ψ) (simp add: primes-psi-def sum-upto-altdef)

end

```

6 Shapiro's Tauberian Theorem

```

theory Shapiro-Tauberian
imports
  More-Dirichlet-Misc
  Prime-Number-Theorem.Prime-Counting-Functions
  Prime-Distribution-Elementary-Library
begin

```

6.1 Proof

Given an arithmetic function $a(n)$, Shapiro's Tauberian theorem relates the sum $\sum_{n \leq x} a(n)$ to the weighted sums $\sum_{n \leq x} a(n) \lfloor \frac{x}{n} \rfloor$ and $\sum_{n \leq x} a(n)/n$. More precisely, it shows that if $\sum_{n \leq x} a(n) \lfloor \frac{x}{n} \rfloor = x \ln x + O(x)$, then:

- $\sum_{n \leq x} \frac{a(n)}{n} = \ln x + O(1)$
- $\sum_{n \leq x} a(n) \leq Bx$ for some constant $B \geq 0$ and all $x \geq 0$
- $\sum_{n \leq x} a(n) \geq Cx$ for some constant $C > 0$ and all $x \geq 1/C$

```

locale shapiro-tauberian =
  fixes  $a :: nat \Rightarrow real$  and  $A S T :: real \Rightarrow real$ 
  defines  $A \equiv sum-upto (\lambda n. a n / n)$ 
  defines  $S \equiv sum-upto a$ 
  defines  $T \equiv (\lambda x. dirichlet-prod' a floor x)$ 
  assumes  $a\text{-nonneg}$ :  $\bigwedge n. n > 0 \implies a n \geq 0$ 
  assumes  $a\text{-asymptotics}$ :  $(\lambda x. T x - x * \ln x) \in O(\lambda x. x)$ 
begin

lemma fin: finite X if  $X \subseteq \{n. real n \leq x\}$  for  $X x$ 
  by (rule finite-subset[of -  $\{..nat [x]\}$ ]) (use that in  $\langle auto simp: le-nat-iff le-floor-iff \rangle$ )

lemma S-mono:  $S x \leq S y$  if  $x \leq y$  for  $x y$ 
  unfolding  $S\text{-def}$   $sum-upto\text{-def}$  using that by (intro sum-mono2 fin[of - y] a-nonneg)
  auto

lemma split:
  fixes  $f :: nat \Rightarrow real$ 
  assumes  $\alpha \in \{0..1\}$ 
  shows  $sum-upto f x = sum-upto f (\alpha * x) + (\sum n \mid n > 0 \wedge real n \in \{\alpha * x < .. x\}. f n)$ 
proof (cases  $x > 0$ )
  case False
  hence *:  $\{n. n > 0 \wedge real n \leq x\} = \{\}$   $\{n. n > 0 \wedge real n \in \{\alpha * x < .. x\}\} = \{\}$ 
    using mult-right-mono[of  $\alpha 1 x$ ] assms by auto
  have  $\alpha * x \leq 0$ 
    using False assms by (intro mult-nonneg-nonpos) auto
  hence **:  $\{n. n > 0 \wedge real n \leq \alpha * x\} = \{\}$ 
    by auto
  show ?thesis
    unfolding  $sum-upto\text{-def}$  * ** by auto
next
  case True
  have  $sum-upto f x = (\sum n \mid n > 0 \wedge real n \leq x. f n)$ 
    by (simp add: sum-upto-def)
  also have  $\{n. n > 0 \wedge real n \leq x\} =$ 
     $\{n. n > 0 \wedge real n \leq \alpha * x\} \cup \{n. n > 0 \wedge real n \in \{\alpha * x < .. x\}\}$ 
    using assms True mult-right-mono[of  $\alpha 1 x$ ] by (force intro: order-trans)
  also have  $(\sum n \in \dots. f n) = sum-upto f (\alpha * x) + (\sum n \mid n > 0 \wedge real n \in \{\alpha * x < .. x\}. f n)$ 
    by (subst sum.union-disjoint) (auto intro: fin simp: sum-upto-def)
  finally show ?thesis .
qed

```

lemma *S-diff-T-diff*: $S x - S (x / 2) \leq T x - 2 * T (x / 2)$

proof –

note $fin = fin[of - x]$

have *T-diff-eq*:

$T x - 2 * T (x / 2) = sum-upto (\lambda n. a n * ([x / n] - 2 * [x / (2 * n)])) (x / 2) +$
 $(\sum n \mid n > 0 \wedge real n \in \{x/2 <..x\}. a n * [x / n])$

unfolding *T-def dirichlet-prod'-def*

by (*subst split[where $\alpha = 1/2$]*)

 (*simp-all add: sum-upto-def sum-subtractf ring-distrib*
 sum-distrib-left sum-distrib-right mult-ac)

have $S x - S (x / 2) = (\sum n \mid n > 0 \wedge real n \in \{x/2 <..x\}. a n)$

unfolding *S-def by (subst split[where $\alpha = 1/2$]) (auto simp: sum-upto-def)*

also have $\dots = (\sum n \mid n > 0 \wedge real n \in \{x/2 <..x\}. a n * [x / n])$

proof (*intro sum.cong*)

fix n **assume** $n \in \{n. n > 0 \wedge real n \in \{x/2 <..x\}\}$

hence $x / n \geq 1$ $x / n < 2$ **by** (*auto simp: field-simps*)

hence $[x / n] = 1$ **by** *linarith*

thus $a n = a n * [x / n]$ **by** *simp*

qed *auto*

also have $\dots = 0 + \dots$ **by** *simp*

also have $0 \leq sum-upto (\lambda n. a n * ([x / n] - 2 * [x / (2 * n)])) (x / 2)$

unfolding *sum-upto-def*

proof (*intro sum-nonneg mult-nonneg-nonneg a-nonneg*)

fix n **assume** $n \in \{n. n > 0 \wedge real n \leq x / 2\}$

hence $x / real n \geq 2$ **by** (*auto simp: field-simps*)

thus $real-of-int ([x / n] - 2 * [x / (2 * n)]) \geq 0$

using *le-mult-floor[of 2 x / (2 * n)] by (simp add: mult-ac)*

qed *auto*

also have $\dots + (\sum n \mid n > 0 \wedge real n \in \{x/2 <..x\}. a n * [x / n]) = T x - 2$

$* T (x / 2)$

using *T-diff-eq ..*

finally show $S x - S (x / 2) \leq T x - 2 * T (x / 2)$ **by** *simp*

qed

lemma

shows *diff-bound-strong*: $\exists c \geq 0. \forall x \geq 0. x * A x - T x \in \{0..c*x\}$

and *asymptotics*: $(\lambda x. A x - \ln x) \in O(\lambda. 1)$

and *upper*: $\exists c \geq 0. \forall x \geq 0. S x \leq c * x$

and *lower*: $\exists c > 0. \forall x \geq 1/c. S x \geq c * x$

and *bitheta*: $S \in \Theta(\lambda x. x)$

proof –

– We first prove the third case, i. e. the upper bound for S .

have $(\lambda x. S x - S (x / 2)) \in O(\lambda x. T x - 2 * T (x / 2))$

proof (*rule le-imp-bigo-real*)

show *eventually* $(\lambda x. S x - S (x / 2) \geq 0)$ *at-top*

using *eventually-ge-at-top[of 0]*

```

proof eventually-elim
  case (elim x)
  thus ?case using S-mono[of x / 2 x] by simp
qed
next
  show eventually ( $\lambda x. S x - S (x / 2) \leq 1 * (T x - 2 * T (x / 2))$ ) at-top
  using S-diff-T-diff by simp
qed auto
also have ( $\lambda x. T x - 2 * T (x / 2) \in O(\lambda x. x)$ )
proof -
  have ( $\lambda x. T x - 2 * T (x / 2) =$ 
    ( $\lambda x. (T x - x * \ln x) - 2 * (T (x / 2) - (x / 2) * \ln (x / 2))$ 
    +  $x * (\ln x - \ln (x / 2))$ )) by (simp add: algebra-simps)
  also have ...  $\in O(\lambda x. x)$ 
proof (rule sum-in-bigo, rule sum-in-bigo)
  show ( $\lambda x. T x - x * \ln x \in O(\lambda x. x)$ ) by (rule a-asymptotics)
next
  have ( $\lambda x. T (x / 2) - (x / 2) * \ln (x / 2) \in O(\lambda x. x / 2)$ )
  using a-asymptotics by (rule landau-o.big.compose) real-asymp+
  thus ( $\lambda x. 2 * (T (x / 2) - x / 2 * \ln (x / 2)) \in O(\lambda x. x)$ )
  unfolding cmult-in-bigo-iff by (subst (asm) landau-o.big.cdiv) auto
qed real-asymp+
finally show ?thesis .
qed
finally have S-diff-bigo: ( $\lambda x. S x - S (x / 2) \in O(\lambda x. x)$ ) .

```

obtain *c1* **where** *c1*: $c1 \geq 0 \wedge x. x \geq 0 \implies S x \leq c1 * x$

```

proof -
from S-diff-bigo have ( $\lambda n. S (\text{real } n) - S (\text{real } n / 2) \in O(\lambda n. \text{real } n)$ )
by (rule landau-o.big.compose) real-asymp
from natfun-bigoE[OF this, of 1] obtain c
  where  $c > 0 \forall n \geq 1. |S (\text{real } n) - S (\text{real } n / 2)| \leq c * \text{real } n$  by auto
hence  $c: S (\text{real } n) - S (\text{real } n / 2) \leq c * \text{real } n$  if  $n \geq 1$  for  $n$ 
  using S-mono[of real n 2 * real n] that by auto
have c-twoPow:  $S (2 \wedge \text{Suc } n / 2) - S (2 \wedge n / 2) \leq c * 2 \wedge n$  for  $n$ 
  using c[of 2 ^ n] by simp

```

have *S-twoPow-le*: $S (2 \wedge k) \leq 2 * c * 2 \wedge k$ **for** k

```

proof -
  have [simp]:  $\{0 <.. \text{Suc } 0\} = \{1\}$  by auto
  have ( $\sum r < \text{Suc } k. S (2 \wedge \text{Suc } r / 2) - S (2 \wedge r / 2) \leq (\sum r < \text{Suc } k. c * 2$ 
 $\wedge r$ )
    by (intro sum-mono c-twoPow)
  also have ( $\sum r < \text{Suc } k. S (2 \wedge \text{Suc } r / 2) - S (2 \wedge r / 2) = S (2 \wedge k)$ )
  by (subst sum-lessThan-telescope) (auto simp: S-def sum-upto-altdef)
  also have ( $\sum r < \text{Suc } k. c * 2 \wedge r = c * (\sum r < \text{Suc } k. 2 \wedge r)$ )
  unfolding sum-distrib-left ..
  also have ( $\sum r < \text{Suc } k. 2 \wedge r :: \text{real} = 2 \wedge \text{Suc } k - 1$ )
  by (subst geometric-sum) auto

```



```

also have  $c * \dots \leq c * 2^{\wedge} \text{Suc } k$ 
  using  $\langle c > 0 \rangle$  by (intro mult-left-mono) auto
finally show  $S (2^{\wedge} k) \leq 2 * c * 2^{\wedge} k$  by simp
qed

have S-le:  $S x \leq 4 * c * x$  if  $x \geq 0$  for  $x$ 
proof (cases  $x \geq 1$ )
  case False
    with that have  $x \in \{0..<1\}$  by auto
    thus ?thesis using  $\langle c > 0 \rangle$  by (auto simp: S-def sum-upto-altdef)
  next
    case True
      hence  $x: x \geq 1$  by simp
      define  $n$  where  $n = \text{nat } \lfloor \log 2 x \rfloor$ 
      have  $2^{\text{pow real } n} \leq 2^{\text{pow } (\log 2 x)}$ 
        unfolding n-def using  $x$  by (intro powr-mono) auto
      hence ge:  $2^{\wedge} n \leq x$  using  $x$  by (subst (asm) powr-realpow) auto

      have  $2^{\text{pow real } (\text{Suc } n)} > 2^{\text{pow } (\log 2 x)}$ 
        unfolding n-def using  $x$  by (intro powr-less-mono) linarith+
      hence less:  $2^{\wedge} (\text{Suc } n) > x$  using  $x$  by (subst (asm) powr-realpow) auto

      have  $S x \leq S (2^{\wedge} \text{Suc } n)$ 
        using less by (intro S-mono) auto
      also have  $\dots \leq 2 * c * 2^{\wedge} \text{Suc } n$ 
        by (intro S-twopow-le)
      also have  $\dots = 4 * c * 2^{\wedge} n$ 
        by simp
      also have  $\dots \leq 4 * c * x$ 
        by (intro mult-left-mono ge) (use  $x \langle c > 0 \rangle$  in auto)
      finally show  $S x \leq 4 * c * x$  .
qed

```

```

with that[of  $4 * c$ ] and  $\langle c > 0 \rangle$  show ?thesis by auto
qed
thus  $\exists c \geq 0. \forall x \geq 0. S x \leq c * x$  by auto

```

— The asymptotics of A follows from this immediately:

```

have a-strong:  $x * A x - T x \in \{0..c1 * x\}$  if  $x: x \geq 0$  for  $x$ 
proof –
  have sum-upto  $(\lambda n. a n * \text{frac } (x / n)) x \leq \text{sum-upto } (\lambda n. a n * 1) x$  unfolding
sum-upto-def
    by (intro sum-mono mult-left-mono a-nonneg) (auto intro: less-imp-le frac-lt-1)
  also have  $\dots = S x$  unfolding S-def by simp
  also from  $x$  have  $\dots \leq c1 * x$  by (rule c1)
  finally have sum-upto  $(\lambda n. a n * \text{frac } (x / n)) x \leq c1 * x$  .
  moreover have sum-upto  $(\lambda n. a n * \text{frac } (x / n)) x \geq 0$ 
    unfolding sum-upto-def by (intro sum-nonneg mult-nonneg-nonneg a-nonneg)
auto

```

ultimately have $\text{sum-upto } (\lambda n. a n * \text{frac } (x / n)) x \in \{0..c1*x\}$ **by** *auto*
also have $\text{sum-upto } (\lambda n. a n * \text{frac } (x / n)) x = x * A x - T x$
by (*simp add: T-def A-def sum-upto-def sum-subtractf frac-def algebra-simps*
sum-distrib-left sum-distrib-right dirichlet-prod'-def)
finally show *?thesis* .
qed
thus $\exists c \geq 0. \forall x \geq 0. x * A x - T x \in \{0..c*x\}$
using $\langle c1 \geq 0 \rangle$ **by** (*intro exI[of - c1]*) *auto*
hence $(\lambda x. x * A x - T x) \in O(\lambda x. x)$
using *a-strong* $\langle c1 \geq 0 \rangle$
by (*intro le-imp-bigo-real[of c1] eventually-mono[OF eventually-ge-at-top[of 1]]*)
auto
from this and a-asymptotics have $(\lambda x. (x * A x - T x) + (T x - x * \ln x)) \in$
 $O(\lambda x. x)$
by (*rule sum-in-bigo*)
hence $(\lambda x. x * (A x - \ln x)) \in O(\lambda x. x * 1)$
by (*simp add: algebra-simps*)
thus *bigo*: $(\lambda x. A x - \ln x) \in O(\lambda x. 1)$
by (*subst (asm) landau-o.big.mult-cancel-left*) *auto*

— It remains to show the lower bound for S .

define R **where** $R = (\lambda x. A x - \ln x)$
obtain M **where** $M: \bigwedge x. x \geq 1 \implies |R x| \leq M$
proof –
have $(\lambda n. R (\text{real } n)) \in O(\lambda-. 1)$
using *bigo unfolding R-def* **by** (*rule landau-o.big.compose*) *real-asymp*
from *natfun-bigoE[OF this, of 0]* **obtain** M **where** $M: M > 0 \bigwedge n. |R (\text{real } n)| \leq M$
by *auto*

have $|R x| \leq M + \ln 2$ **if** $x: x \geq 1$ **for** x

proof –

define n **where** $n = \text{nat } \lfloor x \rfloor$
have $|R x - R (\text{real } n)| = \ln (x / n)$
using x **by** (*simp add: R-def A-def sum-upto-altdef n-def ln-div*)
also {
have $x \leq \text{real } n + 1$
unfolding *n-def* **by** *linarith*
also have $1 \leq \text{real } n$
using x **unfolding** *n-def* **by** *simp*
finally have $\ln (x / n) \leq \ln 2$
using x **by** (*simp add: field-simps*)
}
finally have $|R x| \leq |R (\text{real } n)| + \ln 2$
by *linarith*
also have $|R (\text{real } n)| \leq M$
by (*rule M*)
finally show $|R x| \leq M + \ln 2$ **by** *simp*
qed

with *that*[of $M + \ln 2$] **show** *?thesis* **by** *blast*
qed
have $M \geq 0$ **using** M [of 1] **by** *simp*

have *A-diff-ge*: $A x - A (\alpha * x) \geq -\ln \alpha - 2 * M$
if $\alpha: \alpha \in \{0 < .. < 1\}$ **and** $x \geq 1 / \alpha$ **for** $x \alpha :: \text{real}$
proof –
from *that* **have** $1 < \text{inverse } \alpha * 1$ **by** (*simp add: field-simps*)
also **have** $\dots \leq \text{inverse } \alpha * (\alpha * x)$
using $\langle x \geq 1 / \alpha \rangle$ **and** α **by** (*intro mult-left-mono*) (*auto simp: field-simps*)
also **from** α **have** $\dots = x$ **by** *simp*
finally **have** $x > 1$.
note $x = \text{this } \langle x \rangle \geq 1 / \alpha$

have $-\ln \alpha - M - M \leq -\ln \alpha - |R x| - |R (\alpha * x)|$
using $x \alpha$ **by** (*intro diff-mono M*) (*auto simp: field-simps*)
also **have** $\dots \leq -\ln \alpha + R x - R (\alpha * x)$
by *linarith*
also **have** $\dots = A x - A (\alpha * x)$
using αx **by** (*simp add: R-def ln-mult*)
finally **show** $A x - A (\alpha * x) \geq -\ln \alpha - 2 * M$ **by** *simp*
qed

define α **where** $\alpha = \exp (-2 * M - 1)$
have $\alpha \in \{0 < .. < 1\}$
using $\langle M \geq 0 \rangle$ **by** (*auto simp: alpha-def*)

have *S-ge*: $S x \geq \alpha * x$ **if** $x: x \geq 1 / \alpha$ **for** x
proof –
have $1 = -\ln \alpha - 2 * M$
by (*simp add: alpha-def*)
also **have** $\dots \leq A x - A (\alpha * x)$
by (*intro A-diff-ge*) *fact+*
also **have** $\dots = (\sum n \mid n > 0 \wedge \text{real } n \in \{\alpha * x < .. x\}. a n / n)$
unfolding *A-def* **using** $\langle \alpha \in \{0 < .. < 1\} \rangle$ **by** (*subst split*[**where** $\alpha = \alpha$]) *auto*
also **have** $\dots \leq (\sum n \mid n > 0 \wedge \text{real } n \in \{\alpha * x < .. x\}. a n / (\alpha * x))$
using $x \langle \alpha \in \{0 < .. < 1\} \rangle$ **by** (*intro sum-mono divide-left-mono a-nonneg*) *auto*
also **have** $\dots = (\sum n \mid n > 0 \wedge \text{real } n \in \{\alpha * x < .. x\}. a n) / (\alpha * x)$
by (*simp add: sum-divide-distrib*)
also **have** $\dots \leq S x / (\alpha * x)$
using $x \langle \alpha \in \{0 < .. < 1\} \rangle$ **unfolding** *S-def sum-upto-def*
by (*intro divide-right-mono sum-mono2 a-nonneg*) (*auto simp: field-simps*)
finally **show** $S x \geq \alpha * x$
using $\langle \alpha \in \{0 < .. < 1\} \rangle x$ **by** (*simp add: field-simps*)
qed

thus $\exists c > 0. \forall x \geq 1 / c. S x \geq c * x$
using $\langle \alpha \in \{0 < .. < 1\} \rangle$ **by** (*intro exI*[of $-\alpha$]) *auto*

have *S-nonneg*: $S x \geq 0$ **for** x

```

    unfolding S-def sum-upto-def by (intro sum-nonneg a-nonneg) auto
  have eventually ( $\lambda x. |S x| \geq \alpha * |x|$ ) at-top
    using eventually-ge-at-top[of max 0 (1 /  $\alpha$ )]
  proof eventually-elim
    case (elim x)
    with S-ge[of x] elim show ?case by auto
  qed
  hence  $S \in \Omega(\lambda x. x)$ 
    using  $\langle \alpha \in \{0 < \cdot < 1\} \rangle$  by (intro landau-omega.bigI[of  $\alpha$ ]) auto
  moreover have  $S \in O(\lambda x. x)$ 
  proof (intro bigO eventually-mono[OF eventually-ge-at-top[of 0]])
    fix x :: real assume  $x \geq 0$ 
    thus  $\text{norm } (S x) \leq c1 * \text{norm } x$ 
      using c1(2)[of x] by (auto simp: S-nonneg)
  qed
  ultimately show  $S \in \Theta(\lambda x. x)$ 
    by (intro bithetaI)
  qed
end

```

6.2 Applications to the Chebyshev functions

We can now apply Shapiro's Tauberian theorem to ψ and ϑ .

lemma *dirichlet-prod-mangoldt1-floor-bigo*:

includes *prime-counting-notation*

shows ($\lambda x. \text{dirichlet-prod}' (\lambda n. \text{ind prime } n * \ln n) \text{ floor } x - x * \ln x$) $\in O(\lambda x. x)$

proof –

— This is a perhaps somewhat roundabout way of proving this statement. We show this using the asymptotics of \mathfrak{M} : $\mathfrak{M}(x) = \ln x + O(1)$

We proved this before (which was a bit of work, but not that much). Apostol, on the other hand, shows the following statement first and then deduces the asymptotics of \mathfrak{M} with Shapiro's Tauberian theorem instead. This might save a bit of work, but it is probably negligible.

define *R* where $R = (\lambda x. \text{sum-upto } (\lambda i. \text{ind prime } i * \ln i * \text{frac } (x / i)) x)$

have *: $R x \in \{0.. \ln 4 * x\}$ **if** $x \geq 1$ **for** *x*

proof –

have $R x \leq \vartheta x$

unfolding *R-def prime-sum-upto-altdef1 sum-upto-def ϑ -def*

by (*intro sum-mono*) (*auto simp: ind-def less-imp-le*[OF *frac-lt-1*] *dest!*: *prime-gt-1-nat*)

also have $\dots < \ln 4 * x$

by (*rule ϑ -upper-bound*) *fact+*

finally have $R x \leq \ln 4 * x$ **by** *auto*

moreover have $R x \geq 0$ **unfolding** *R-def sum-upto-def*

by (*intro sum-nonneg mult-nonneg-nonneg*) (*auto simp: ind-def*)

ultimately show *?thesis* **by** *auto*

qed

have *eventually* $(\lambda x. |R x| \leq \ln 4 * |x|)$ *at-top*
using *eventually-ge-at-top*[of 1] **by** *eventually-elim* (*use * in auto*)
hence $R \in O(\lambda x. x)$ **by** (*intro landau-o.bigI*[of $\ln 4$]) *auto*

have $(\lambda x. \text{dirichlet-prod}' (\lambda n. \text{ind prime } n * \ln n) \text{floor } x - x * \ln x) =$
 $(\lambda x. x * (\mathfrak{M} x - \ln x) - R x)$
by (*auto simp: primes-M-def dirichlet-prod'-def prime-sum-upto-altdef1 sum-upto-def*
frac-def sum-subtractf sum-distrib-left sum-distrib-right algebra-simps
R-def)
also have $\dots \in O(\lambda x. x)$
proof (*rule sum-in-bigo*)
have $(\lambda x. x * (\mathfrak{M} x - \ln x)) \in O(\lambda x. x * 1)$
by (*intro landau-o.big.mult mertens-bounded*) *auto*
thus $(\lambda x. x * (\mathfrak{M} x - \ln x)) \in O(\lambda x. x)$ **by** *simp*
qed *fact+*
finally show *?thesis* .
qed

lemma *dirichlet-prod'-mangoldt-floor-asymptotics*:
 $(\lambda x. \text{dirichlet-prod}' \text{mangoldt floor } x - x * \ln x + x) \in O(\ln)$
proof –
have *dirichlet-prod' mangoldt floor* = $(\lambda x. \text{sum-upto } \ln x)$
unfolding *sum-upto-ln-conv-sum-upto-mangoldt dirichlet-prod'-def*
by (*intro sum-upto-cong' ext*) *auto*
hence $(\lambda x. \text{dirichlet-prod}' \text{mangoldt floor } x - x * \ln x + x) = (\lambda x. \text{sum-upto } \ln$
 $x - x * \ln x + x)$
by *simp*
also have $\dots \in O(\ln)$
by (*rule sum-upto-ln-stirling-weak-bigo*)
finally show $(\lambda x. \text{dirichlet-prod}' \text{mangoldt } (\lambda x. \text{real-of-int } \lfloor x \rfloor) x - x * \ln x +$
 $x) \in O(\ln)$.
qed

interpretation ψ : *shapiro-tauberian mangoldt sum-upto* $(\lambda n. \text{mangoldt } n / n)$
primes-psi
dirichlet-prod' mangoldt floor
proof *unfold-locales*
have *dirichlet-prod' mangoldt floor* = $(\lambda x. \text{sum-upto } \ln x)$
unfolding *sum-upto-ln-conv-sum-upto-mangoldt dirichlet-prod'-def*
by (*intro sum-upto-cong' ext*) *auto*
hence $(\lambda x. \text{dirichlet-prod}' \text{mangoldt floor } x - x * \ln x + x) = (\lambda x. \text{sum-upto } \ln$
 $x - x * \ln x + x)$
by *simp*
also have $\dots \in O(\ln)$
by (*rule sum-upto-ln-stirling-weak-bigo*)
also have $\ln \in O(\lambda x::\text{real}. x)$ **by** *real-asymp*
finally have $(\lambda x. \text{dirichlet-prod}' \text{mangoldt } (\lambda x. \text{real-of-int } \lfloor x \rfloor) x - x * \ln x + x$
 $- x)$

$\in O(\lambda x. x)$ **by** (*rule sum-in-bigo*) *auto*
thus ($\lambda x. \text{dirichlet-prod}' \text{ mangoldt } (\lambda x. \text{real-of-int } \lfloor x \rfloor) x - x * \ln x$) $\in O(\lambda x. x)$
by *simp*
qed (*simp-all add: primes-psi-def mangoldt-nonneg*)

thm $\psi.\text{asymptotics } \psi.\text{upper } \psi.\text{lower}$

interpretation ϑ : *shapiro-tauberian* $\lambda n. \text{ind prime } n * \ln n$
sum-upto ($\lambda n. \text{ind prime } n * \ln n / n$) *primes-theta dirichlet-prod'* ($\lambda n. \text{ind prime } n * \ln n$) *floor*
proof *unfold-locales*
fix $n :: \text{nat}$ **show** $\text{ind prime } n * \ln n \geq 0$
by (*auto simp: ind-def dest: prime-gt-1-nat*)
next
show ($\lambda x. \text{dirichlet-prod}' (\lambda n. \text{ind prime } n * \ln n) \text{ floor } x - x * \ln x$) $\in O(\lambda x. x)$
by (*rule dirichlet-prod-mangoldt1-floor-bigo*)
qed (*simp-all add: primes-theta-def mangoldt-nonneg prime-sum-upto-altdef1 [abs-def]*)

thm $\vartheta.\text{asymptotics } \vartheta.\text{upper } \vartheta.\text{lower}$

lemma *sum-upto-psi-x-over-n-asymptotics*:
 $(\lambda x. \text{sum-upto } (\lambda n. \text{primes-psi } (x / n)) x - x * \ln x + x) \in O(\ln)$
and *sum-upto-psi-x-over-n-asymptotics*:
 $(\lambda x. \text{sum-upto } (\lambda n. \text{primes-theta } (x / n)) x - x * \ln x) \in O(\lambda x. x)$
using *dirichlet-prod-mangoldt1-floor-bigo dirichlet-prod'-mangoldt-floor-asymptotics*
by (*simp-all add: dirichlet-prod'-floor-conv-sum-upto primes-theta-def primes-psi-def prime-sum-upto-altdef1*)

end

7 Bounds on partial sums of the ζ function

theory *Partial-Zeta-Bounds*
imports
Euler-MacLaurin.Euler-MacLaurin-Landau
Zeta-Function.Zeta-Function
Prime-Number-Theorem.Prime-Number-Theorem-Library
Prime-Distribution-Elementary-Library
begin

We employ Euler–MacLaurin’s summation formula to obtain asymptotic estimates for the partial sums of the Riemann $\zeta(s)$ function for fixed real a ,

i. e. the function

$$f(n) = \sum_{k=1}^n k^{-s} .$$

We distinguish various cases. The case $s = 1$ is simply the Harmonic numbers and is treated apart from the others.

lemma *harm-asymp-equiv: sum-upto* $(\lambda n. 1 / n) \sim[at-top] \ln$

proof –

have *sum-upto* $(\lambda n. n \text{ powr } -1) \sim[at-top] (\lambda x. \ln (\lfloor x \rfloor + 1))$

proof (*rule asymp-equiv-sandwich*)

have *eventually* $(\lambda x. \text{sum-upto } (\lambda n. n \text{ powr } -1) x \in \{\ln (\lfloor x \rfloor + 1).. \ln \lfloor x \rfloor + 1\}) \text{ at-top}$

using *eventually-ge-at-top*[of 1]

proof *eventually-elim*

case (*elim x*)

have *sum-upto* $(\lambda n. \text{real } n \text{ powr } -1) x = \text{harm } (\text{nat } \lfloor x \rfloor)$

unfolding *sum-upto-altdef harm-def* **by** (*intro sum.cong*) (*auto simp: field-simps powr-minus*)

also have $\dots \in \{\ln (\lfloor x \rfloor + 1).. \ln \lfloor x \rfloor + 1\}$

using *elim harm-le*[of *nat* $\lfloor x \rfloor$] *ln-le-harm*[of *nat* $\lfloor x \rfloor$]

by (*auto simp: le-nat-iff le-floor-iff*)

finally show *?case* **by** *simp*

qed

thus *eventually* $(\lambda x. \text{sum-upto } (\lambda n. n \text{ powr } -1) x \geq \ln (\lfloor x \rfloor + 1)) \text{ at-top}$

eventually $(\lambda x. \text{sum-upto } (\lambda n. n \text{ powr } -1) x \leq \ln \lfloor x \rfloor + 1) \text{ at-top}$

by (*eventually-elim; simp*)+

qed *real-asymp+*

also have $\dots \sim[at-top] \ln$ **by** *real-asymp*

finally show *?thesis* **by** (*simp add: powr-minus field-simps*)

qed

lemma

fixes $s :: \text{real}$

assumes $s : s > 0 \ s \neq 1$

shows *zeta-partial-sum-bigo-pos:*

$(\lambda n. (\sum_{k=1}^n \text{real } k \text{ powr } -s) - \text{real } n \text{ powr } (1 - s) / (1 - s) - \text{Re } (\text{zeta } s))$

$\in O(\lambda x. \text{real } x \text{ powr } -s)$

and *zeta-partial-sum-bigo-pos':*

$(\lambda n. \sum_{k=1}^n \text{real } k \text{ powr } -s) = o$

$(\lambda n. \text{real } n \text{ powr } (1 - s) / (1 - s) + \text{Re } (\text{zeta } s)) + o O(\lambda x. \text{real } x \text{ powr } -s)$

proof –

define F **where** $F = (\lambda x. x \text{ powr } (1 - s) / (1 - s))$

define f **where** $f = (\lambda x. x \text{ powr } -s)$

define f' **where** $f' = (\lambda x. -s * x \text{ powr } (-s-1))$

define z **where** $z = \text{Re } (\text{zeta } s)$

interpret *euler-maclaurin-nat'* $F f (!) [f, f'] 1 0 z \{\}$

```

proof
  have ( $\lambda b. (\sum k=1..b. \text{real } k \text{ powr } -s) - \text{real } b \text{ powr } (1 - s) / (1 - s) - \text{real } b \text{ powr } -s / 2$ )
     $\longrightarrow \text{Re } (\text{zeta } s) - 0$ 
  proof (intro tendsto-diff)
    let ?g =  $\lambda b. (\sum i < b. \text{complex-of-real } (\text{real } i + 1) \text{ powr } - \text{complex-of-real } s)$ 
    -
       $\text{of-nat } b \text{ powr } (1 - \text{complex-of-real } s) / (1 - \text{complex-of-real } s)$ 
    have  $\forall_F b \text{ in at-top. } \text{Re } (?g \ b) = (\sum k=1..b. \text{real } k \text{ powr } -s) - \text{real } b \text{ powr } (1 - s) / (1 - s)$ 
    using eventually-ge-at-top[of 1]
    proof eventually-elim
    case (elim b)
    have  $(\sum k=1..b. \text{real } k \text{ powr } -s) = (\sum k < b. \text{real } (\text{Suc } k) \text{ powr } -s)$ 
    by (intro sum.reindex-bij-witness[of - Suc  $\lambda n. n - 1$ ] auto)
    also have  $\dots - \text{real } b \text{ powr } (1 - s) / (1 - s) = \text{Re } (?g \ b)$ 
    by (auto simp: powr-Reals-eq add-ac)
    finally show ?case ..
  qed
  moreover have ( $\lambda b. \text{Re } (?g \ b)$ )  $\longrightarrow \text{Re } (\text{zeta } s)$ 
  using hurwitz-zeta-critical-strip[of of-real s 1] s
  by (intro tendsto-intros (simp add: zeta-def))
  ultimately show ( $\lambda b. (\sum k=1..b. \text{real } k \text{ powr } -s) - \text{real } b \text{ powr } (1 - s) / (1 - s)$ )  $\longrightarrow \text{Re } (\text{zeta } s)$ 
  by (blast intro: Lim-transform-eventually)
  qed (use s in real-asymp)
  thus ( $\lambda b. (\sum k = 1..b. f \ (\text{real } k)) - F \ (\text{real } b) - (\sum i < 2 * 0 + 1. (\text{bernoulli}' \ (\text{Suc } i) / \text{fact } (\text{Suc } i)) *_R \ ([f, f'] ! i) \ (\text{real } b))$ )
     $\longrightarrow z$  by (simp add: f-def F-def z-def)
  qed (use  $\langle s \neq 1 \rangle$  in
     $\langle$ auto intro!: derivative-eq-intros continuous-intros simp flip: has-field-derivative-iff-has-vector-derivative simp: F-def f-def f'-def nth-Cons split: nat.splits $\rangle$ )
  {
    fix n :: nat assume n:  $n \geq 1$ 
    have  $(\sum k=1..n. \text{real } k \text{ powr } -s) = n \text{ powr } (1 - s) / (1 - s) + z + 1/2 * n \text{ powr } -s - \text{EM-remainder } 1 \ f' \ (\text{int } n)$ 
    using euler-maclaurin-strong-nat'[of n] n by (simp add: F-def f-def)
  } note * = this

  have ( $\lambda n. (\sum k=1..n. \text{real } k \text{ powr } -s) - n \text{ powr } (1 - s) / (1 - s) - z$ )  $\in \Theta(\lambda n. 1/2 * n \text{ powr } -s - \text{EM-remainder } 1 \ f' \ (\text{int } n))$ 
  using * by (intro bigthetaI-cong eventually-mono[OF eventually-ge-at-top[of 1]] auto)
  also have ( $\lambda n. 1/2 * n \text{ powr } -s - \text{EM-remainder } 1 \ f' \ (\text{int } n)$ )  $\in O(\lambda n. n \text{ powr } -s)$ 

```


$-s)$
proof (*intro sum-in-bigo*)
have $(\lambda x. \text{norm } (EM\text{-remainder } 1 f' (int x))) \in O(\lambda x. \text{real } x \text{ powr } -s)$
proof (*rule EM-remainder-strong-bigo-nat* [**where** $a = 1$ **and** $Y = \{\}$])
fix $x :: \text{real}$ **assume** $x \geq 1$
show $\text{norm } (f' x) \leq s * x \text{ powr } (-s-1)$
using s **by** (*simp add: f'-def*)
next
from s **show** $((\lambda x. x \text{ powr } -s) \longrightarrow 0)$ *at-top* **by** *real-asymp*
qed (*auto simp: f'-def intro!: continuous-intros derivative-eq-intros*)
thus $(\lambda x. EM\text{-remainder } 1 f' (int x)) \in O(\lambda x. \text{real } x \text{ powr } -s)$
by *simp*
qed *real-asymp+*
finally show $(\lambda n. (\sum k=1..n. \text{real } k \text{ powr } -s) - \text{real } n \text{ powr } (1 - s) / (1 - s) - z)$
 $\in O(\lambda x. \text{real } x \text{ powr } -s) .$
thus $(\lambda n. \sum k=1..n. \text{real } k \text{ powr } -s) = o$
 $(\lambda n. \text{real } n \text{ powr } (1 - s) / (1 - s) + z) + o O(\lambda x. \text{real } x \text{ powr } -s)$
by (*subst set-minus-plus [symmetric]*) (*simp-all add: fun-diff-def algebra-simps*)
qed

lemma *zeta-tail-bigo*:
fixes $s :: \text{real}$
assumes $s: s > 1$
shows $(\lambda n. \text{Re } (hurwitz\text{-zeta } (real\ n + 1) s)) \in O(\lambda x. \text{real } x \text{ powr } (1 - s))$
proof $-$
have [*simp*]: *complex-of-real* $(\text{Re } (zeta\ s)) = zeta\ s$
using *zeta-real[of s]* **by** (*auto elim!: Reals-cases*)

from s **have** $s': s > 0\ s \neq 1$ **by** *auto*
have $(\lambda n. -\text{Re } (hurwitz\text{-zeta } (real\ n + 1) s) - \text{real } n \text{ powr } (1 - s) / (1 - s) + \text{real } n \text{ powr } (1 - s) / (1 - s)) \in O(\lambda x. \text{real } x \text{ powr } (1 - s))$
proof (*rule sum-in-bigo*)
have $(\lambda n. -\text{Re } (hurwitz\text{-zeta } (real\ n + 1) s) - \text{real } n \text{ powr } (1 - s) / (1 - s)) =$
 $(\lambda n. (\sum k=1..n. \text{real } k \text{ powr } -s) - \text{real } n \text{ powr } (1 - s) / (1 - s) - \text{Re } (zeta\ s))$
 $(\text{is } ?lhs = ?rhs)$
proof
fix $n :: \text{nat}$
have $hurwitz\text{-zeta } (1 + \text{real } n) s = zeta\ s - (\sum k < n. \text{real } (Suc\ k) \text{ powr } -s)$
by (*subst hurwitz-zeta-shift*) (*use assms in <auto simp: zeta-def powr-Reals-eq>*)
also have $(\sum k < n. \text{real } (Suc\ k) \text{ powr } -s) = (\sum k=1..n. \text{real } k \text{ powr } -s)$
by (*rule sum.reindex-bij-witness[of - λk. k - 1 Suc]*) *auto*
finally show $?lhs\ n = ?rhs\ n$ **by** (*simp add: add-ac*)
qed
also have $\dots \in O(\lambda x. \text{real } x \text{ powr } (-s))$
by (*rule zeta-partial-sum-bigo-pos*) (*use s in auto*)

also have $(\lambda x. \text{real } x \text{ powr } (-s)) \in O(\lambda x. \text{real } x \text{ powr } (1-s))$
by *real-asymp*
finally show $(\lambda n. -\text{Re } (\text{hurwitz-zeta } (\text{real } n + 1) s) - \text{real } n \text{ powr } (1 - s) / (1 - s)) \in \dots$
qed (*use s in real-asymp*)
thus *?thesis by simp*
qed

lemma *zeta-tail-bigo'*:

fixes $s :: \text{real}$
assumes $s : s > 1$
shows $(\lambda n. \text{Re } (\text{hurwitz-zeta } (\text{real } n) s)) \in O(\lambda x. \text{real } x \text{ powr } (1 - s))$
proof –
have $(\lambda n. \text{Re } (\text{hurwitz-zeta } (\text{real } n) s)) \in \Theta(\lambda n. \text{Re } (\text{hurwitz-zeta } (\text{real } (n - 1) + 1) s))$
by (*intro bigthetaI-cong eventually-mono[OF eventually-ge-at-top[of 1]]*)
(auto simp: of-nat-diff)
also have $(\lambda n. \text{Re } (\text{hurwitz-zeta } (\text{real } (n - 1) + 1) s)) \in O(\lambda x. \text{real } (x - 1) \text{ powr } (1 - s))$
by (*rule landau-o.big.compose[OF zeta-tail-bigo[OF assms]]*) *real-asymp*
also have $(\lambda x. \text{real } (x - 1) \text{ powr } (1 - s)) \in O(\lambda x. \text{real } x \text{ powr } (1 - s))$
by *real-asymp*
finally show *?thesis* .
qed

lemma

fixes $s :: \text{real}$
assumes $s : s > 0$
shows *zeta-partial-sum-bigo-neg*:
 $(\lambda n. (\sum_{i=1..n} \text{real } i \text{ powr } s) - n \text{ powr } (1 + s) / (1 + s)) \in O(\lambda n. n \text{ powr } s)$
and *zeta-partial-sum-bigo-neg'*:
 $(\lambda n. (\sum_{i=1..n} \text{real } i \text{ powr } s)) = o(\lambda n. n \text{ powr } (1 + s) / (1 + s)) + o(O(\lambda n. n \text{ powr } s))$
proof –
define F **where** $F = (\lambda x. x \text{ powr } (1 + s) / (1 + s))$
define f **where** $f = (\lambda x. x \text{ powr } s)$
define f' **where** $f' = (\lambda x. s * x \text{ powr } (s - 1))$

have $(\sum_{i=1..n} f (\text{real } i)) - F n = 1 / 2 - F 1 + f n / 2 + \text{EM-remainder}' 1 f' 1 (\text{real } n)$ **if** $n : n \geq 1$ **for** n

proof –

have $(\sum_{i \in \{1 <.. n\}} f (\text{real } i)) - \text{integral } \{\text{real } 1.. \text{real } n\} f =$
 $(\sum_{k < 1.} (\text{bernoulli}' (\text{Suc } k) / \text{fact } (\text{Suc } k)) *_{\mathbb{R}} (([f, f'] ! k) (\text{real } n) - ([f, f'] ! k) (\text{real } 1))) + \text{EM-remainder}' 1 ([f, f'] ! 1) (\text{real } 1) (\text{real } n)$
by (*rule euler-maclaurin-strong-raw-nat[where Y = {}]*)
(use <s > 0> <n ≥ 1> in
<auto intro!: derivative-eq-intros continuous-intros
simp flip: has-field-derivative-iff-has-vector-derivative

simp: F-def f-def f'-def nth-Cons split: nat.splits)

also have $(\sum_{i \in \{1 <..n\}} f(\text{real } i)) = (\sum_{i \in \text{insert } 1 \{1 <..n\}} f(\text{real } i)) - f$
 1

using n **by** *(subst sum.insert) auto*

also from n **have** $\text{insert } 1 \{1 <..n\} = \{1..n\}$ **by** *auto*

finally have $(\sum_{i=1..n} f(\text{real } i)) - F n = f 1 + (\text{integral } \{1..real\ } n\} f - F$
 $n) +$

$(f(\text{real } n) - f 1) / 2 + \text{EM-remainder}' 1 f' 1 (\text{real } n)$ **by** *simp*

hence $(\sum_{i=1..n} f(\text{real } i)) - F n = 1 / 2 + (\text{integral } \{1..real\ } n\} f - F n)$
 $+$

$f(\text{real } n) / 2 + \text{EM-remainder}' 1 f' 1 (\text{real } n)$

using s **by** *(simp add: f-def diff-divide-distrib)*

also have $(f \text{ has-integral } (F(\text{real } n) - F 1)) \{1..real\ } n$ **using** *assms n*

by *(intro fundamental-theorem-of-calculus)*

(auto simp flip: has-field-derivative-iff-has-vector-derivative simp: F-def f-def
intro!: derivative-eq-intros continuous-intros)

hence $\text{integral } \{1..real\ } n\} f - F n = - F 1$

by *(simp add: has-integral-iff)*

also have $1 / 2 + (-F 1) + f(\text{real } n) / 2 = 1 / 2 - F 1 + f n / 2$

by *simp*

finally show *?thesis .*

qed

hence $(\lambda n. (\sum_{i=1..n} f(\text{real } i)) - F n) \in$
 $\Theta(\lambda n. 1 / 2 - F 1 + f n / 2 + \text{EM-remainder}' 1 f' 1 (\text{real } n))$

by *(intro bighetaI-cong eventually-mono[OF eventually-ge-at-top[of 1]])*

also have $(\lambda n. 1 / 2 - F 1 + f n / 2 + \text{EM-remainder}' 1 f' 1 (\text{real } n))$
 $\in O(\lambda n. \text{real } n \text{ powr } s)$

unfolding *F-def f-def*

proof *(intro sum-in-bigo)*

have $(\lambda x. \text{integral } \{1..real\ } x\} (\lambda t. \text{pbernpoly } 1 t *_R f' t)) \in O(\lambda n. 1 / s * \text{real}$
 $n \text{ powr } s)$

proof *(intro landau-o.big.compose[OF integral-bigo])*

have $(\lambda x. \text{pbernpoly } 1 x *_R f' x) \in O(\lambda x. 1 * x \text{ powr } (s - 1))$

by *(intro landau-o.big.mult pbernpoly-bigo) (auto simp: f'-def)*

thus $(\lambda x. \text{pbernpoly } 1 x *_R f' x) \in O(\lambda x. x \text{ powr } (s - 1))$

by *simp*

from s **show** *filterlim* $(\lambda a. 1 / s * a \text{ powr } s)$ *at-top at-top* **by** *real-asymp*

next

fix $a' x :: \text{real}$ **assume** $a' \geq 1 \ a' \leq x$

thus $(\lambda a. \text{pbernpoly } 1 a *_R f' a) \text{ integrable-on } \{a'..x\}$

by *(intro integrable-EM-remainder') (auto intro!: continuous-intros simp:*
 $f'-\text{def})$

qed *(use s in <auto intro!: filterlim-real-sequentially continuous-intros derivative-eq-intros>)*

thus $(\lambda x. \text{EM-remainder}' 1 f' 1 (\text{real } x)) \in O(\lambda n. \text{real } n \text{ powr } s)$

using $\langle s > 0 \rangle$ **by** *(simp add: EM-remainder'-def)*

qed *(use <s > 0 in real-asymp)+*

finally show $(\lambda n. (\sum_{i=1..n} \text{real } i \text{ powr } s) - n \text{ powr } (1 + s) / (1 + s)) \in$

$O(\lambda n. n \text{ powr } s)$
by (*simp add: f-def F-def*)
thus $(\lambda n. (\sum_{i=1..n} \text{real } i \text{ powr } s)) = o (\lambda n. n \text{ powr } (1 + s) / (1 + s)) + o$
 $O(\lambda n. n \text{ powr } s)$
by (*subst set-minus-plus [symmetric]*) (*simp-all add: fun-diff-def algebra-simps*)
qed

lemma *zeta-partial-sum-le-pos*:

assumes $s > 0 \ s \neq 1$
defines $z \equiv \text{Re } (\text{zeta } (\text{complex-of-real } s))$
shows $\exists c > 0. \forall x \geq 1. |\text{sum-upto } (\lambda n. n \text{ powr } -s) \ x - (x \text{ powr } (1-s) / (1-s) + z)| \leq c * x \text{ powr } -s$
proof (*rule sum-upto-asymptotics-lift-nat-real*)
show $(\lambda n. (\sum_{k=1..n} \text{real } k \text{ powr } -s) - (\text{real } n \text{ powr } (1 - s) / (1 - s) + z))$
 $\in O(\lambda n. \text{real } n \text{ powr } -s)$
using *zeta-partial-sum-bigo-pos[OF assms(1,2)]* **unfolding** *z-def*
by (*simp add: algebra-simps*)

from *assms* **have** $s < 1 \vee s > 1$ **by** *linarith*

thus $(\lambda n. \text{real } n \text{ powr } (1 - s) / (1 - s) + z - (\text{real } (\text{Suc } n) \text{ powr } (1 - s) / (1 - s) + z))$
 $\in O(\lambda n. \text{real } n \text{ powr } -s)$

by *standard* (*use* $\langle s > 0 \rangle$ **in** $\langle \text{real-asympt} \rangle$)

show $(\lambda n. \text{real } n \text{ powr } -s) \in O(\lambda n. \text{real } (\text{Suc } n) \text{ powr } -s)$

by *real-asympt*

show *mono-on* $(\lambda a. a \text{ powr } -s) \{1..\} \vee \text{mono-on } (\lambda x. - (x \text{ powr } -s)) \{1..\}$

using *assms* **by** (*intro disjI2*) (*auto intro!: mono-onI powr-mono2'*)

from *assms* **have** $s < 1 \vee s > 1$ **by** *linarith*

hence *mono-on* $(\lambda a. a \text{ powr } (1 - s) / (1 - s) + z) \{1..\}$

proof

assume $s < 1$

thus *?thesis* **using** $\langle s > 0 \rangle$

by (*intro mono-onI powr-mono2 divide-right-mono add-right-mono*) *auto*

next

assume $s > 1$

thus *?thesis*

by (*intro mono-onI le-imp-neg-le add-right-mono divide-right-mono-neg powr-mono2'*)

auto

qed

thus *mono-on* $(\lambda a. a \text{ powr } (1 - s) / (1 - s) + z) \{1..\} \vee$
mono-on $(\lambda x. - (x \text{ powr } (1 - s) / (1 - s) + z)) \{1..\}$ **by** *blast*

qed *auto*

lemma *zeta-partial-sum-le-pos'*:

assumes $s > 0 \ s \neq 1$

defines $z \equiv \text{Re } (\text{zeta } (\text{complex-of-real } s))$

shows $\exists c > 0. \forall x \geq 1. |\text{sum-upto } (\lambda n. n \text{ powr } -s) \ x - x \text{ powr } (1-s) / (1-s)|$

$\leq c$
proof –
have $\exists c > 0. \forall x \geq 1. |sum\text{-upto } (\lambda n. n \text{ powr } -s) x - x \text{ powr } (1-s) / (1-s)| \leq c$
 $* 1$
proof (rule *sum-upto-asymptotics-lift-nat-real*)
have $(\lambda n. (\sum k = 1..n. real\ k \text{ powr } -s) - (real\ n \text{ powr } (1-s) / (1-s) + z))$
 $\in O(\lambda n. real\ n \text{ powr } -s)$
using *zeta-partial-sum-bigo-pos* [*OF assms(1,2)*] **unfolding** *z-def*
by (*simp add: algebra-simps*)
also have $(\lambda n. real\ n \text{ powr } -s) \in O(\lambda n. 1)$
using *assms* **by** *real-asymp*
finally have $(\lambda n. (\sum k = 1..n. real\ k \text{ powr } -s) - real\ n \text{ powr } (1-s) / (1-s) - z)$
 $\in O(\lambda n. 1)$ **by** (*simp add: algebra-simps*)
hence $(\lambda n. (\sum k = 1..n. real\ k \text{ powr } -s) - real\ n \text{ powr } (1-s) / (1-s) - z + z) \in O(\lambda n. 1)$
by (*rule sum-in-bigo*) *auto*
thus $(\lambda n. (\sum k = 1..n. real\ k \text{ powr } -s) - (real\ n \text{ powr } (1-s) / (1-s))) \in O(\lambda n. 1)$
by *simp*

from *assms* **have** $s < 1 \vee s > 1$ **by** *linarith*
thus $(\lambda n. real\ n \text{ powr } (1-s) / (1-s) - (real\ (Suc\ n) \text{ powr } (1-s) / (1-s))) \in O(\lambda n. 1)$
by *standard* (use $\langle s > 0 \rangle$ **in** $\langle real\text{-asymp+} \rangle$)
show *mono-on* $(\lambda a. 1) \{1..\}$ \vee *mono-on* $(\lambda x::real. -1 :: real) \{1..\}$
using *assms* **by** (*intro disjI2*) (*auto intro!: mono-onI powr-mono2'*)

from *assms* **have** $s < 1 \vee s > 1$ **by** *linarith*
hence *mono-on* $(\lambda a. a \text{ powr } (1-s) / (1-s)) \{1..\}$
proof
assume $s < 1$
thus *?thesis* **using** $\langle s > 0 \rangle$
by (*intro mono-onI powr-mono2 divide-right-mono add-right-mono*) *auto*
next
assume $s > 1$
thus *?thesis*
by (*intro mono-onI le-imp-neg-le add-right-mono divide-right-mono-neg powr-mono2'*) *auto*
qed
thus *mono-on* $(\lambda a. a \text{ powr } (1-s) / (1-s)) \{1..\}$ \vee
mono-on $(\lambda x. - (x \text{ powr } (1-s) / (1-s))) \{1..\}$ **by** *blast*
qed *auto*
thus *?thesis* **by** *simp*
qed

lemma *zeta-partial-sum-le-pos''*:
assumes $s > 0$ $s \neq 1$

shows $\exists c > 0. \forall x \geq 1. |\text{sum-upto } (\lambda n. n \text{ powr } -s) x| \leq c * x \text{ powr } \max 0 (1 - s)$
proof –
from *zeta-partial-sum-le-pos'*[*OF assms*] **obtain** *c* **where**
c: $c > 0 \wedge x. x \geq 1 \implies |\text{sum-upto } (\lambda x. \text{real } x \text{ powr } -s) x - x \text{ powr } (1 - s) / (1 - s)| \leq c$
by *auto*

{
fix *x* :: *real* **assume** *x*: $x \geq 1$
have $|\text{sum-upto } (\lambda x. \text{real } x \text{ powr } -s) x| \leq |x \text{ powr } (1 - s) / (1 - s)| + c$
using *c(1) c(2)*[*OF x*] **by** *linarith*
also have $|x \text{ powr } (1 - s) / (1 - s)| = x \text{ powr } (1 - s) / |1 - s|$
using *assms* **by** *simp*
also have $\dots \leq x \text{ powr } \max 0 (1 - s) / |1 - s|$
using *x* **by** (*intro divide-right-mono powr-mono*) *auto*
also have $c = c * x \text{ powr } 0$ **using** *x* **by** *simp*
also have $c * x \text{ powr } 0 \leq c * x \text{ powr } \max 0 (1 - s)$
using *c(1) x* **by** (*intro mult-left-mono powr-mono*) *auto*
also have $x \text{ powr } \max 0 (1 - s) / |1 - s| + c * x \text{ powr } \max 0 (1 - s) =$
 $(1 / |1 - s| + c) * x \text{ powr } \max 0 (1 - s)$
by (*simp add: algebra-simps*)
finally have $|\text{sum-upto } (\lambda x. \text{real } x \text{ powr } -s) x| \leq (1 / |1 - s| + c) * x \text{ powr } \max 0 (1 - s)$
by *simp*
}
moreover have $(1 / |1 - s| + c) > 0$
using *c assms* **by** (*intro add-pos-pos divide-pos-pos*) *auto*
ultimately show *?thesis* **by** *blast*
qed

lemma *zeta-partial-sum-le-pos-bigo*:
assumes $s > 0 \ s \neq 1$
shows $(\lambda x. \text{sum-upto } (\lambda n. n \text{ powr } -s) x) \in O(\lambda x. x \text{ powr } \max 0 (1 - s))$
proof –
from *zeta-partial-sum-le-pos''*[*OF assms*] **obtain** *c*
where $\forall x \geq 1. |\text{sum-upto } (\lambda n. n \text{ powr } -s) x| \leq c * x \text{ powr } \max 0 (1 - s)$ **by** *auto*
thus *?thesis*
by (*intro bigO[of - c] eventually-mono[OF eventually-ge-at-top[of 1]]*) *auto*
qed

lemma *zeta-partial-sum-01-asympt-equiv*:
assumes $s \in \{0 < .. < 1\}$
shows $\text{sum-upto } (\lambda n. n \text{ powr } -s) \sim[\text{at-top}] (\lambda x. x \text{ powr } (1 - s) / (1 - s))$
proof –
from *zeta-partial-sum-le-pos'*[*of s*] *assms* **obtain** *c* **where**
c: $c > 0 \ \forall x \geq 1. |\text{sum-upto } (\lambda x. \text{real } x \text{ powr } -s) x - x \text{ powr } (1 - s) / (1 - s)| \leq c$ **by** *auto*

hence $(\lambda x. \text{sum-upto } (\lambda x. \text{real } x \text{ powr } -s) x - x \text{ powr } (1 - s) / (1 - s)) \in O(\lambda-. 1)$
by $(\text{intro } \text{bigoI}[\text{of } - c] \text{ eventually-mono}[\text{OF eventually-ge-at-top}[\text{of } 1]]) \text{ auto}$
also have $(\lambda-. 1) \in o(\lambda x. x \text{ powr } (1 - s) / (1 - s))$
using $\text{assms by real-asymp}$
finally show $?thesis$
by $(\text{rule } \text{smallo-imp-asymp-equiv})$
qed

lemma $\text{zeta-partial-sum-gt-1-asymp-equiv}$:

fixes $s :: \text{real}$
assumes $s > 1$
defines $\zeta \equiv \text{Re } (\text{zeta } s)$
shows $\text{sum-upto } (\lambda n. n \text{ powr } -s) \sim[\text{at-top}] (\lambda x. \zeta)$
proof $-$
have $[\text{simp}]: \zeta \neq 0$
using $\text{assms zeta-Re-gt-1-nonzero}[\text{of } s] \text{ zeta-real}[\text{of } s] \text{ by } (\text{auto elim!}: \text{Reals-cases})$
from $\text{zeta-partial-sum-le-pos}[\text{of } s] \text{ assms}$ **obtain** $c \text{ where}$
 $c: c > 0 \ \forall x \geq 1. |\text{sum-upto } (\lambda x. \text{real } x \text{ powr } -s) x - (x \text{ powr } (1 - s) / (1 - s) + \zeta)| \leq$
 $c * x \text{ powr } -s \text{ by } (\text{auto simp}: \zeta\text{-def})$
hence $(\lambda x. \text{sum-upto } (\lambda x. \text{real } x \text{ powr } -s) x - \zeta - x \text{ powr } (1 - s) / (1 - s)) \in O(\lambda x. x \text{ powr } -s)$
by $(\text{intro } \text{bigoI}[\text{of } - c] \text{ eventually-mono}[\text{OF eventually-ge-at-top}[\text{of } 1]]) \text{ auto}$
also have $(\lambda x. x \text{ powr } -s) \in o(\lambda-. 1)$
using $\langle s > 1 \rangle \text{ by real-asymp}$
finally have $(\lambda x. \text{sum-upto } (\lambda x. \text{real } x \text{ powr } -s) x - \zeta - x \text{ powr } (1 - s) / (1 - s) +$
 $x \text{ powr } (1 - s) / (1 - s)) \in o(\lambda-. 1)$
by $(\text{rule } \text{sum-in-smallo}) (\text{use } \langle s > 1 \rangle \text{ in real-asymp})$
thus $?thesis \text{ by } (\text{simp add}: \text{smallo-imp-asymp-equiv})$
qed

lemma $\text{zeta-partial-sum-pos-bigtheta}$:

assumes $s > 0 \ s \neq 1$
shows $\text{sum-upto } (\lambda n. n \text{ powr } -s) \in \Theta(\lambda x. x \text{ powr } \max 0 (1 - s))$
proof $(\text{cases } s > 1)$
case False
thus $?thesis$
using $\text{asymp-equiv-imp-bigtheta}[\text{OF zeta-partial-sum-01-asymp-equiv}[\text{of } s]] \text{ assms}$
by $(\text{simp add}: \text{max-def})$
next
case True
have $[\text{simp}]: \text{Re } (\text{zeta } s) \neq 0$
using $\text{True zeta-Re-gt-1-nonzero}[\text{of } s] \text{ zeta-real}[\text{of } s] \text{ by } (\text{auto elim!}: \text{Reals-cases})$
show $?thesis$
using $\text{True asymp-equiv-imp-bigtheta}[\text{OF zeta-partial-sum-gt-1-asymp-equiv}[\text{of } s]]$

by (*simp add: max-def*)
qed

lemma *zeta-partial-sum-le-neg*:

assumes $s > 0$

shows $\exists c > 0. \forall x \geq 1. |\text{sum-upto } (\lambda n. n \text{ powr } s) x - x \text{ powr } (1 + s) / (1 + s)| \leq c * x \text{ powr } s$

proof (*rule sum-upto-asymptotics-lift-nat-real*)

show $(\lambda n. (\sum k = 1..n. \text{real } k \text{ powr } s) - (\text{real } n \text{ powr } (1 + s) / (1 + s))) \in O(\lambda n. \text{real } n \text{ powr } s)$

using *zeta-partial-sum-bigo-neg[OF assms(1)]* **by** (*simp add: algebra-simps*)

show $(\lambda n. \text{real } n \text{ powr } (1 + s) / (1 + s) - (\text{real } (\text{Suc } n) \text{ powr } (1 + s) / (1 + s))) \in O(\lambda n. \text{real } n \text{ powr } s)$

using *assms* **by** *real-asymp*

show $(\lambda n. \text{real } n \text{ powr } s) \in O(\lambda n. \text{real } (\text{Suc } n) \text{ powr } s)$

by *real-asymp*

show *mono-on* $(\lambda a. a \text{ powr } s) \{1..\} \vee \text{mono-on } (\lambda x. - (x \text{ powr } s)) \{1..\}$

using *assms* **by** (*intro disjI1*) (*auto intro!: mono-onI powr-mono2*)

show *mono-on* $(\lambda a. a \text{ powr } (1 + s) / (1 + s)) \{1..\} \vee \text{mono-on } (\lambda x. - (x \text{ powr } (1 + s) / (1 + s))) \{1..\}$

using *assms* **by** (*intro disjI1 divide-right-mono powr-mono2 mono-onI*) *auto*

qed *auto*

lemma *zeta-partial-sum-neg-asymp-equiv*:

assumes $s > 0$

shows $\text{sum-upto } (\lambda n. n \text{ powr } s) \sim[\text{at-top}] (\lambda x. x \text{ powr } (1 + s) / (1 + s))$

proof –

from *zeta-partial-sum-le-neg[of s]* *assms* **obtain** *c* **where**

$c: c > 0 \forall x \geq 1. |\text{sum-upto } (\lambda x. \text{real } x \text{ powr } s) x - x \text{ powr } (1 + s) / (1 + s)| \leq c * x \text{ powr } s$

by *auto*

hence $(\lambda x. \text{sum-upto } (\lambda x. \text{real } x \text{ powr } s) x - x \text{ powr } (1 + s) / (1 + s)) \in O(\lambda x. x \text{ powr } s)$

by (*intro bigoI[of - c] eventually-mono[OF eventually-ge-at-top[of 1]]*) *auto*

also have $(\lambda x. x \text{ powr } s) \in o(\lambda x. x \text{ powr } (1 + s) / (1 + s))$

using *assms* **by** *real-asymp*

finally show *?thesis*

by (*rule smallo-imp-asymp-equiv*)

qed

end

8 The summatory Möbius μ function

theory *Moebius-Mu-Sum*

imports

More-Dirichlet-Misc

Dirichlet-Series.Partial-Summation
Prime-Number-Theorem.Prime-Counting-Functions
Dirichlet-Series.Arithmetic-Summatory-Asymptotics
Shapiro-Tauberian
Partial-Zeta-Bounds
Prime-Number-Theorem.Prime-Number-Theorem-Library
Prime-Distribution-Elementary-Library

begin

In this section, we shall examine the summatory Möbius μ function $M(x) := \sum_{n \leq x} \mu(n)$. The main result is that $M(x) \in o(x)$ is equivalent to the Prime Number Theorem.

context

includes *prime-counting-notation*
fixes $M H :: \text{real} \Rightarrow \text{real}$
defines $M \equiv \text{sum-upto moebius-mu}$
defines $H \equiv \text{sum-upto } (\lambda n. \text{moebius-mu } n * \ln n)$

begin

lemma *sum-upto-moebius-mu-integral*: $x > 1 \implies ((\lambda t. M t / t) \text{ has-integral } M x * \ln x - H x) \{1..x\}$

and *sum-upto-moebius-mu-integrable*: $a \geq 1 \implies (\lambda t. M t / t) \text{ integrable-on } \{a..b\}$

proof –

{
fix $a b :: \text{real}$
assume $ab: a \geq 1 a < b$
have $((\lambda t. M t * (1 / t)) \text{ has-integral } M b * \ln b - M a * \ln a -$
 $(\sum n \in \text{real} - \{a<..b\}. \text{moebius-mu } n * \ln (\text{real } n))) \{a..b\}$
unfolding *M-def* **using** ab
by (*intro partial-summation-strong* [**where** $X = \{\}$])
 $(\text{auto intro!}: \text{derivative-eq-intros continuous-intros}$
 $\text{simp flip}: \text{has-field-derivative-iff-has-vector-derivative})$
} note $* = \text{this}$
{
fix $x :: \text{real}$ **assume** $x: x > 1$
have $(\sum n \in \text{real} - \{1<..x\}. \text{moebius-mu } n * \ln (\text{real } n)) = H x$
unfolding *H-def sum-upto-def* **by** (*intro sum.mono-neutral-cong-left*) (*use x*
in *auto*)
thus $((\lambda t. M t / t) \text{ has-integral } M x * \ln x - H x) \{1..x\}$ **using** $*[\text{of } 1 x]$ x **by**
simp
}
{
fix $a b :: \text{real}$ **assume** $ab: a \geq 1$
show $(\lambda t. M t / t) \text{ integrable-on } \{a..b\}$
using $*[\text{of } a b]$ ab
by (*cases a b rule: linorder-cases*) (*auto intro: integrable-negligible*)
}
qed

lemma *sum-moebius-mu-bound*:

assumes $x \geq 0$

shows $|M x| \leq x$

proof –

have $|M x| \leq \text{sum-upto } (\lambda n. |\text{moebius-mu } n|) x$

unfolding *M-def sum-upto-def* **by** (*rule sum-abs*)

also have $\dots \leq \text{sum-upto } (\lambda n. 1) x$

unfolding *sum-upto-def* **by** (*intro sum-mono*) (*auto simp: moebius-mu-def*)

also have $\dots \leq x$ **using** *assms*

by (*simp add: sum-upto-altdef*)

finally show *?thesis* .

qed

lemma *sum-moebius-mu-aux1*: $(\lambda x. M x / x - H x / (x * \ln x)) \in O(\lambda x. 1 / \ln x)$

proof –

define *R* **where** $R = (\lambda x. \text{integral } \{1..x\} (\lambda t. M t / t))$

have *eventually* $(\lambda x. M x / x - H x / (x * \ln x) = R x / (x * \ln x))$ *at-top*

using *eventually-gt-at-top[of 1]*

proof *eventually-elim*

case (*elim x*)

thus *?case*

using *sum-upto-moebius-mu-integral[of x]* **by** (*simp add: R-def has-integral-iff field-simps*)

qed

hence $(\lambda x. M x / x - H x / (x * \ln x)) \in \Theta(\lambda x. R x / (x * \ln x))$

by (*intro bigthetaI-cong*)

also have $(\lambda x. R x / (x * \ln x)) \in O(\lambda x. x / (x * \ln x))$

proof (*intro landau-o.big.divide-right*)

have $M \in O(\lambda x. x)$

using *sum-moebius-mu-bound*

by (*intro bigOI[where c = 1] eventually-mono[OF eventually-ge-at-top[of 0]]*)

auto

hence $(\lambda t. M t / t) \in O(\lambda t. 1)$

by (*simp add: landau-divide-simps*)

thus $R \in O(\lambda x. x)$

unfolding *R-def*

by (*intro integral-bigo[where g' = λ-. 1]*)

(*auto simp: filterlim-ident has-integral-iff intro!: sum-upto-moebius-mu-integrable*)

qed (*intro eventually-mono[OF eventually-gt-at-top[of 1]], auto*)

also have $(\lambda x::\text{real}. x / (x * \ln x)) \in \Theta(\lambda x. 1 / \ln x)$

by *real-asymp*

finally show *?thesis* .

qed

lemma *sum-moebius-mu-aux2*: $((\lambda x. M x / x - H x / (x * \ln x)) \longrightarrow 0)$ *at-top*

proof –

have $(\lambda x. M x / x - H x / (x * \ln x)) \in O(\lambda x. 1 / \ln x)$

by (rule sum-moebius-mu-aux1)
 also have $(\lambda x. 1 / \ln x) \in o(\lambda \cdot. 1 :: \text{real})$
 by real-asymp
 finally show ?thesis by (auto dest!: smalloD-tendsto)
 qed

lemma sum-moebius-mu-ln-eq: $H = (\lambda x. - \text{dirichlet-prod}' \text{ moebius-mu } \psi x)$

proof

fix $x :: \text{real}$
 have $\text{fds mangoldt} = (\text{fds-deriv } (\text{fds moebius-mu}) * \text{fds-zeta} :: \text{real fds})$
 using $\text{fds-mangoldt}'$ by (simp add: mult-ac)
 hence eq: $\text{fds-deriv } (\text{fds moebius-mu}) = \text{fds moebius-mu} * (\text{fds mangoldt} :: \text{real fds})$
 by (subst (asm) fds-moebius-inversion [symmetric])
 have $-H x = \text{sum-upto } (\lambda n. -\ln n * \text{moebius-mu } n) x$
 by (simp add: H-def sum-upto-def sum-negf mult-ac)
 also have $\dots = \text{sum-upto } (\lambda n. \text{dirichlet-prod } \text{moebius-mu } \text{mangoldt } n) x$
 using eq by (intro sum-upto-cong) (auto simp: fds-eq-iff fds-nth-deriv fds-nth-mult)
 also have $\dots = \text{dirichlet-prod}' \text{ moebius-mu } \psi x$
 by (subst sum-upto-dirichlet-prod) (simp add: primes-psi-def dirichlet-prod'-def)
 finally show $H x = -\text{dirichlet-prod}' \text{ moebius-mu } \psi x$
 by simp

qed

theorem PNT-implies-sum-moebius-mu-sublinear:

assumes $\psi \sim[at-top] (\lambda x. x)$
 shows $M \in o(\lambda x. x)$
 proof -
 have $((\lambda x. H x / (x * \ln x)) \longrightarrow 0) \text{ at-top}$
 proof (rule tendstoI)
 fix $\varepsilon' :: \text{real}$ assume $\varepsilon': \varepsilon' > 0$
 define ε where $\varepsilon = \varepsilon' / 2$
 from ε' have $\varepsilon: \varepsilon > 0$ by (simp add: ε -def)
 from assms have $((\lambda x. \psi x / x) \longrightarrow 1) \text{ at-top}$
 by (elim asymp-equivD-strong) (auto intro!: eventually-mono[OF eventually-gt-at-top[of 0]])
 from tendstoD[OF this ε] have eventually $(\lambda x. |\psi x / x - 1| < \varepsilon) \text{ at-top}$
 by (simp add: dist-norm)
 hence eventually $(\lambda x. |\psi x - x| < \varepsilon * x) \text{ at-top}$
 using eventually-gt-at-top[of 0] by eventually-elim (auto simp: abs-if field-simps)
 then obtain A' where $A': \bigwedge x. x \geq A' \implies |\psi x - x| < \varepsilon * x$
 by (auto simp: eventually-at-top-linorder)
 define A where $A = \max 2 A'$
 from A' have $A: A \geq 2 \bigwedge x. x \geq A \implies |\psi x - x| < \varepsilon * x$
 by (auto simp: A-def)

have H-bound: $|H x| / (x * \ln x) \leq (1 + \varepsilon + \psi A) / \ln x + \varepsilon$ if $x \geq A$ for x

proof -

```

from  $\langle x \geq A \rangle$  have  $x \geq 2$  using  $A(1)$  by linarith
note  $x = \langle x \geq A \rangle \langle x \geq 2 \rangle$ 

define  $y$  where  $y = \text{nat } \lfloor \text{floor } (x / A) \rfloor$ 
have  $\text{real } y = \text{real-of-int } \lfloor x / A \rfloor$  using  $A$   $x$  by (simp add: y-def)
also have  $\text{real-of-int } \lfloor x / A \rfloor \leq x / A$  by linarith
also have  $\dots \leq x$  using  $x$   $A(1)$  by (simp add: field-simps)
finally have  $y \leq x$  .
have  $y \geq 1$  using  $x$   $A(1)$  by (auto simp: y-def le-nat-iff le-floor-iff)
note  $y = \langle y \geq 1 \rangle \langle y \leq x \rangle$ 

define  $S1$  where [simp]:  $S1 = \text{sum-upto } (\lambda m. \text{moebius-mu } m * \psi (x / m))$   $y$ 
define  $S2$  where [simp]:  $S2 = (\sum m \mid m > y \wedge \text{real } m \leq x. \text{moebius-mu } m$ 
 $* \psi (x / m))$ 

have  $\text{fin: finite } \{m. y < m \wedge \text{real } m \leq x\}$ 
by (rule finite-subset[of - {..nat } \lfloor x \rfloor]) (auto simp: le-nat-iff le-floor-iff)
have  $H x = -\text{dirichlet-prod}' \text{moebius-mu } \psi x$ 
by (simp add: sum-moebius-mu-ln-eq)
also have  $\text{dirichlet-prod}' \text{moebius-mu } \psi x =$ 
 $(\sum m \mid m > 0 \wedge \text{real } m \leq x. \text{moebius-mu } m * \psi (x / m))$ 
unfolding dirichlet-prod'-def sum-upto-def ..
also have  $\{m. m > 0 \wedge \text{real } m \leq x\} = \{0 < ..y\} \cup \{m. y < m \wedge \text{real } m \leq x\}$ 
using  $x$   $y$   $A(1)$  by auto
also have  $(\sum m \in \dots. \text{moebius-mu } m * \psi (x / m)) = S1 + S2$ 
unfolding dirichlet-prod'-def sum-upto-def S1-def S2-def using fin
by (subst sum.union-disjoint) (auto intro: sum.cong)
finally have  $\text{abs-}H\text{-eq: } |H x| = |S1 + S2|$  by simp

define  $S1-1$  where [simp]:  $S1-1 = \text{sum-upto } (\lambda m. \text{moebius-mu } m / m)$   $y$ 
define  $S1-2$  where [simp]:  $S1-2 = \text{sum-upto } (\lambda m. \text{moebius-mu } m * (\psi (x /$ 
 $m) - x / m))$   $y$ 

have  $|S1| = |x * S1-1 + S1-2|$ 
by (simp add: sum-upto-def sum-distrib-left sum-distrib-right
mult-ac sum-subtractf ring-distrib)
also have  $\dots \leq x * |S1-1| + |S1-2|$ 
by (rule order.trans[OF abs-triangle-ineq]) (use x in <simp add: abs-mult>)
also have  $\dots \leq x * 1 + \varepsilon * x * (\ln x + 1)$ 
proof (intro add-mono mult-left-mono)
show  $|S1-1| \leq 1$ 
using abs-sum-upto-moebius-mu-over-n-le[of y] by simp
next
have  $|S1-2| \leq \text{sum-upto } (\lambda m. |\text{moebius-mu } m * (\psi (x / m) - x / m)|)$   $y$ 
unfolding S1-2-def sum-upto-def by (rule sum-abs)
also have  $\dots \leq \text{sum-upto } (\lambda m. 1 * (\varepsilon * (x / m)))$   $y$ 
unfolding abs-mult sum-upto-def
proof (intro sum-mono mult-mono less-imp-le[OF A(2)])
fix  $m$  assume  $m: m \in \{i. 0 < i \wedge \text{real } i \leq \text{real } y\}$ 

```

hence $real\ m \leq real\ y$ by *simp*
 also from $x\ A(1)$ have $\dots = of-int\ \lfloor x / A \rfloor$ by (*simp add: y-def*)
 also have $\dots \leq x / A$ by *linarith*
 finally show $A \leq x / real\ m$ using $A(1)\ m$ by (*simp add: field-simps*)
 qed (*auto simp: moebius-mu-def field-simps*)
 also have $\dots = \varepsilon * x * (\sum i \in \{0 <.. y\}. inverse\ (real\ i))$
 by (*simp add: sum-upto-altdef sum-distrib-left divide-simps*)
 also have $(\sum i \in \{0 <.. y\}. inverse\ (real\ i)) = harm\ (nat\ \lfloor y \rfloor)$
 unfolding *harm-def* by (*intro sum.cong*) *auto*
 also have $\dots \leq \ln\ (nat\ \lfloor y \rfloor) + 1$
 by (*rule harm-le*) (*use y in auto*)
 also have $\ln\ (nat\ \lfloor y \rfloor) \leq \ln\ x$
 using y by *simp*
 finally show $|S1-2| \leq \varepsilon * x * (\ln\ x + 1)$ using $\varepsilon\ x$ by *simp*
 qed (*use x in auto*)
 finally have $S1-bound: |S1| \leq x + \varepsilon * x * \ln\ x + \varepsilon * x$
 by (*simp add: algebra-simps*)

have $|S2| \leq (\sum m \mid y < m \wedge real\ m \leq x. |moebius-mu\ m * \psi\ (x / m)|)$
 unfolding *S2-def* by (*rule sum-abs*)
 also have $\dots \leq (\sum m \mid y < m \wedge real\ m \leq x. 1 * \psi\ A)$
 unfolding *abs-mult* using y
 proof (*intro sum-mono mult-mono*)
 fix m assume $m: m \in \{m. y < m \wedge real\ m \leq x\}$
 hence $y < m$ by *simp*
 moreover have $y = of-int\ \lfloor x / A \rfloor$ using $x\ A(1)$ by (*simp add: y-def*)
 ultimately have $\lfloor x / A \rfloor < m$ by *simp*
 hence $x / A \leq real\ m$ by *linarith*
 hence $\psi\ (x / real\ m) \leq \psi\ A$
 using $m\ A(1)$ by (*intro psi-mono*) (*auto simp: field-simps*)
 thus $|\psi\ (x / real\ m)| \leq \psi\ A$
 by (*simp add: psi-nonneg*)
 qed (*auto simp: moebius-mu-def psi-nonneg field-simps intro!: psi-mono*)
 also have $\dots \leq sum-upto\ (\lambda-. 1 * \psi\ A)\ x$
 unfolding *sum-upto-def* by (*intro sum-mono2*) *auto*
 also have $\dots = real\ (nat\ \lfloor x \rfloor) * \psi\ A$
 by (*simp add: sum-upto-altdef*)
 also have $\dots \leq x * \psi\ A$
 using x by (*intro mult-right-mono*) *auto*
 finally have $S2-bound: |S2| \leq x * \psi\ A$.

have $|H\ x| \leq |S1| + |S2|$ using *abs-H-eq* by *linarith*
 also have $\dots \leq x + \varepsilon * x * \ln\ x + \varepsilon * x + x * \psi\ A$
 by (*intro add-mono S1-bound S2-bound*)
 finally have $|H\ x| \leq (1 + \varepsilon + \psi\ A) * x + \varepsilon * x * \ln\ x$
 by (*simp add: algebra-simps*)
 thus $|H\ x| / (x * \ln\ x) \leq (1 + \varepsilon + \psi\ A) / \ln\ x + \varepsilon$
 using x by (*simp add: field-simps*)
 qed

have *eventually* $(\lambda x. |H x| / (x * \ln x) \leq (1 + \varepsilon + \psi A) / \ln x + \varepsilon)$ *at-top*
using *eventually-ge-at-top*[of A] **by** *eventually-elim* (*use* H -*bound* **in** *auto*)
moreover have *eventually* $(\lambda x. (1 + \varepsilon + \psi A) / \ln x + \varepsilon < \varepsilon')$ *at-top*
unfolding ε -*def* **using** ε' **by** *real-asymp*
moreover have *eventually* $(\lambda x. |H x| / (x * \ln x) = |H x / (x * \ln x)|)$ *at-top*
using *eventually-gt-at-top*[of 1] **by** *eventually-elim* (*simp* *add*: *abs-mult*)
ultimately have *eventually* $(\lambda x. |H x / (x * \ln x)| < \varepsilon')$ *at-top*
by *eventually-elim simp*
thus *eventually* $(\lambda x. \text{dist } (H x / (x * \ln x)) \ 0 < \varepsilon')$ *at-top*
by (*simp* *add*: *dist-norm*)
qed
hence $(\lambda x. H x / (x * \ln x)) \in o(\lambda-. \ 1)$
by (*intro* *smalloI-tendsto*) *auto*
hence $(\lambda x. H x / (x * \ln x) + (M x / x - H x / (x * \ln x))) \in o(\lambda-. \ 1)$
proof (*rule* *sum-in-smallo*)
have $(\lambda x. M x / x - H x / (x * \ln x)) \in O(\lambda x. \ 1 / \ln x)$
by (*rule* *sum-moebius-mu-aux1*)
also have $(\lambda x::\text{real}. \ 1 / \ln x) \in o(\lambda-. \ 1)$
by *real-asymp*
finally show $(\lambda x. M x / x - H x / (x * \ln x)) \in o(\lambda-. \ 1)$.
qed
thus *?thesis* **by** (*simp* *add*: *landau-divide-simps*)
qed

theorem *sum-moebius-mu-sublinear-imp-PNT*:

assumes $M \in o(\lambda x. \ x)$
shows $\psi \sim_{[at-top]} (\lambda x. \ x)$

proof –

define $\sigma :: \text{nat} \Rightarrow \text{real}$ **where** [*simp*]: $\sigma = (\lambda n. \ \text{real } (\text{divisor-count } n))$
define C **where** [*simp*]: $C = (\text{euler-mascheroni} :: \text{real})$
define $f :: \text{nat} \Rightarrow \text{real}$ **where** $f = (\lambda n. \ \sigma \ n - \ln \ n - 2 * C)$
define F **where** [*simp*]: $F = \text{sum-upto } f$
write *moebius-mu* (μ)

– The proof is based on the fact that $\psi(x) - x$ can be approximated fairly well by the Dirichlet product $\sum_{n \leq x} \sum_{d|n} \mu(d) f(n/d)$:

have *eq*: $\psi \ x - x = -\text{sum-upto } (\text{dirichlet-prod } \mu \ f) \ x - \text{frac } x - 2 * C$ **if** $x: \ x \geq 1$ **for** x

proof –

have $[x] - \psi \ x - 2 * C =$

$\text{sum-upto } (\lambda-. \ 1) \ x - \text{sum-upto } \text{mangoldt} \ x - \text{sum-upto } (\lambda n. \ \text{if } n = 1 \text{ then } 2 * C \text{ else } 0) \ x$

using x **by** (*simp* *add*: *sum-upto-altdef* ψ -*def* *le-nat-iff* *le-floor-iff*)

also have $\dots = \text{sum-upto } (\lambda n. \ 1 - \text{mangoldt } n - (\text{if } n = 1 \text{ then } 2 * C \text{ else } 0)) \ x$

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

also have $\dots = \text{sum-upto } (\text{dirichlet-prod } \mu \ f) \ x$

by (intro sum-upto-cong refl moebius-inversion)
 (auto simp: divisor-count-def sum-subtractf mangoldt-sum f-def)
finally show $\psi x - x = -\text{sum-upto } (\text{dirichlet-prod } \mu f) x - \text{frac } x - 2 * C$
 by (simp add: algebra-simps frac-def)
qed

— We now obtain a bound of the form $|F x| \leq B * \text{sqrt } x$.

have $F \in O(\text{sqrt})$

proof –

have $F \in \Theta(\lambda x. (\text{sum-upto } \sigma x - (x * \ln x + (2 * C - 1) * x)) -$
 $(\text{sum-upto } \ln x - x * \ln x + x) + 2 * C * \text{frac } x)$ (**is** - $\in \Theta(?rhs)$)

by (intro bigthetaI-cong eventually-mono[OF eventually-ge-at-top[of 1]])

(auto simp: sum-upto-altdef sum-subtractf f-def frac-def algebra-simps

sum.distrib)

also have $?rhs \in O(\text{sqrt})$

proof (rule sum-in-bigo, rule sum-in-bigo)

show $(\lambda x. \text{sum-upto } \sigma x - (x * \ln x + (2 * C - 1) * x)) \in O(\text{sqrt})$

unfolding C-def σ -def **by** (rule summatory-divisor-count-asymptotics)

show $(\lambda x. \text{sum-upto } (\lambda x. \ln (\text{real } x)) x - x * \ln x + x) \in O(\text{sqrt})$

by (rule landau-o.big.trans[OF sum-upto-ln-stirling-weak-bigo]) real-asymp

qed (use euler-mascheroni-pos **in** real-asymp)

finally show $?thesis$.

qed

hence $(\lambda n. F (\text{real } n)) \in O(\text{sqrt})$

by (rule landau-o.big.compose) real-asymp

from natfun-bigoE[OF this, of 1] **obtain** $B :: \text{real}$

where $B: B > 0 \wedge n. n \geq 1 \implies |F (\text{real } n)| \leq B * \text{sqrt } (\text{real } n)$

by auto

have $B': |F x| \leq B * \text{sqrt } x$ **if** $x \geq 1$ **for** x

proof –

have $|F x| \leq B * \text{sqrt } (\text{nat } \lfloor x \rfloor)$

using B(2)[of nat $\lfloor x \rfloor$] **that** **by** (simp add: sum-upto-altdef le-nat-iff le-floor-iff)

also have $\dots \leq B * \text{sqrt } x$

using B(1) **that** **by** (intro mult-left-mono) auto

finally show $?thesis$.

qed

— Next, we obtain a good bound for $\sum_{n \leq x} \frac{1}{\sqrt{n}}$.

from zeta-partial-sum-le-pos''[of 1 / 2] **obtain** A

where $A: A > 0 \wedge x. x \geq 1 \implies |\text{sum-upto } (\lambda n. 1 / \text{sqrt } n) x| \leq A * \text{sqrt } x$

by (auto simp: max-def powr-half-sqrt powr-minus field-simps)

— Finally, we show that $\sum_{n \leq x} \sum_{d|n} \mu(d) f(n/d) \in o(x)$.

have $\text{sum-upto } (\text{dirichlet-prod } \mu f) \in o(\lambda x. x)$

proof (rule landau-o.smallII)

fix $\varepsilon :: \text{real}$

assume $\varepsilon: \varepsilon > 0$

have *: eventually $(\lambda x. |\text{sum-upto } (\text{dirichlet-prod } \mu f) x| \leq \varepsilon * x)$ at-top

if $b: b \geq 1 \ A * B / \text{sqrt } b \leq \varepsilon / 3 \ B / \text{sqrt } b \leq \varepsilon / 3$ **for** b
proof –
define $K :: \text{real}$ **where** $K = \text{sum-upto } (\lambda n. |f n| / n) \ b$
have $C \neq (1 / 2)$ **using** *euler-mascheroni-gt-19-over-33* **by** *auto*
hence $K: K > 0$ **unfolding** *K-def f-def sum-upto-def*
by (*intro sum-pos2[where i = 1]*) (*use <b ≥ 1> in auto*)
have *eventually* $(\lambda x. |M x / x| < \varepsilon / 3 / K)$ *at-top*
using *smalloD-tendsto[OF assms] ε K* **by** (*auto simp: tendsto-iff dist-norm*)
then obtain c' **where** $c': \bigwedge x. x \geq c' \implies |M x / x| < \varepsilon / 3 / K$
by (*auto simp: eventually-at-top-linorder*)
define c **where** $c = \max 1 \ c'$
have $c: |M x| < \varepsilon / 3 / K * x$ **if** $x \geq c$ **for** x
using $c'[of x]$ **that** **by** (*simp add: c-def field-simps*)

show *eventually* $(\lambda x. |\text{sum-upto } (\text{dirichlet-prod } \mu \ f) \ x| \leq \varepsilon * x)$ *at-top*
using *eventually-ge-at-top[of b * c] eventually-ge-at-top[of 1] eventu-*
ally-ge-at-top[of b]
proof *eventually-elim*
case (*elim x*)
define a **where** $a = x / b$
from *elim <b ≥ 1>* **have** $ab: a \geq 1 \ b \geq 1 \ a * b = x$
by (*simp-all add: a-def field-simps*)
from ab **have** $a * 1 \leq a * b$ **by** (*intro mult-mono*) *auto*
hence $a \leq x$ **by** (*simp add: ab(3)*)
from ab **have** $a * 1 \leq a * b$ **and** $1 * b \leq a * b$ **by** (*intro mult-mono;*
simp)**+**
hence $a \leq x \ b \leq x$ **by** (*simp-all add: ab(3)*)
have $a = x / b \ b = x / a$ **using** ab **by** (*simp-all add: field-simps*)

have $\text{sum-upto } (\text{dirichlet-prod } \mu \ f) \ x =$
 $\text{sum-upto } (\lambda n. \mu \ n * F \ (x / n)) \ a + \text{sum-upto } (\lambda n. M \ (x / n) * f \ n)$
 $b - M \ a * F \ b$
unfolding *M-def F-def* **by** (*rule hyperbola-method*) (*use ab in auto*)
also have $|\dots| \leq \varepsilon / 3 * x + \varepsilon / 3 * x + \varepsilon / 3 * x$
proof (*rule order.trans[OF abs-triangle-ineq4] order.trans[OF abs-triangle-ineq]*
add-mono)**+**
have $|\text{sum-upto } (\lambda n. \mu \ n * F \ (x / \text{real } n)) \ a| \leq \text{sum-upto } (\lambda n. |\mu \ n * F$
 $(x / \text{real } n)|) \ a$
unfolding *sum-upto-def* **by** (*rule sum-abs*)
also have $\dots \leq \text{sum-upto } (\lambda n. 1 * (B * \text{sqrt } (x / \text{real } n))) \ a$
unfolding *sum-upto-def abs-mult* **using** $\langle a \leq x \rangle$
by (*intro sum-mono mult-mono B'*) (*auto simp: moebius-mu-def*)
also have $\dots = B * \text{sqrt } x * \text{sum-upto } (\lambda n. 1 / \text{sqrt } n) \ a$
by (*simp add: sum-upto-def sum-distrib-left real-sqrt-divide*)
also have $\dots \leq B * \text{sqrt } x * |\text{sum-upto } (\lambda n. 1 / \text{sqrt } n) \ a|$
using $B(1) \langle x \geq 1 \rangle$ **by** (*intro mult-left-mono*) *auto*
also have $\dots \leq B * \text{sqrt } x * (A * \text{sqrt } a)$
using $\langle a \geq 1 \rangle B(1) \langle x \geq 1 \rangle$ **by** (*intro mult-left-mono A*) *auto*
also have $\dots = A * B / \text{sqrt } b * x$


```

      using ab ⟨x ≥ 1⟩⟨x ≥ 1⟩ by (subst ⟨a = x / b⟩) (simp-all add: field-simps
real-sqrt-divide)
      also have ... ≤ ε / 3 * x using ⟨x ≥ 1⟩ by (intro mult-right-mono b)
auto
      finally show |sum-upto (λn. μ n * F (x / n)) a| ≤ ε / 3 * x .
next
have |sum-upto (λn. M (x / n) * f n) b| ≤ sum-upto (λn. |M (x / n) * f
n|) b
      unfolding sum-upto-def by (rule sum-abs)
      also have ... ≤ sum-upto (λn. ε / 3 / K * (x / n) * |f n|) b
      unfolding sum-upto-def abs-mult
      proof (intro sum-mono mult-right-mono)
        fix n assume n: n ∈ {n. n > 0 ∧ real n ≤ b}
        have c ≥ 0 by (simp add: c-def)
        with n have c * n ≤ c * b by (intro mult-left-mono) auto
        also have ... ≤ x using ⟨b * c ≤ x⟩ by (simp add: algebra-simps)
        finally show |M (x / real n)| ≤ ε / 3 / K * (x / real n)
          by (intro less-imp-le[OF c]) (use n in ⟨auto simp: field-simps⟩)
      qed auto
      also have ... = ε / 3 * x / K * sum-upto (λn. |f n| / n) b
        by (simp add: sum-upto-def sum-distrib-left)
      also have ... = ε / 3 * x
        unfolding K-def [symmetric] using K by simp
      finally show |sum-upto (λn. M (x / real n) * f n) b| ≤ ε / 3 * x .
next
have |M a * F b| ≤ a * (B * sqrt b)
unfolding abs-mult using ab by (intro mult-mono sum-moebius-mu-bound
B') auto
      also have ... = B / sqrt b * x
        using ab(1,2) by (simp add: real-sqrt-mult ⟨b = x / a⟩ real-sqrt-divide
field-simps)
      also have ... ≤ ε / 3 * x using ⟨x ≥ 1⟩ by (intro mult-right-mono b)
auto
      finally show |M a * F b| ≤ ε / 3 * x .
qed
also have ... = ε * x by simp
finally show ?case .
qed
qed

have eventually (λb::real. b ≥ 1 ∧ A * B / sqrt b ≤ ε / 3 ∧ B / sqrt b ≤ ε /
3) at-top
  using ε by (intro eventually-conj; real-asymp)
then obtain b where b ≥ 1 ∧ A * B / sqrt b ≤ ε / 3 ∧ B / sqrt b ≤ ε / 3
  by (auto simp: eventually-at-top-linorder)
from *[OF this] have eventually (λx. |sum-upto (dirichlet-prod μ f) x| ≤ ε *
x) at-top .
thus eventually (λx. norm (sum-upto (dirichlet-prod μ f) x) ≤ ε * norm x)
at-top

```

using *eventually-ge-at-top[of 0]* **by** *eventually-elim simp*
qed
have $(\lambda x. \psi x - x) \in \Theta(\lambda x. -(sum\text{-upto} (dirichlet\text{-prod} \mu f) x + (frac\ x + 2 * C)))$
by (*intro bighetaI-cong eventually-mono[OF eventually-ge-at-top[of 1]]*, *subst eq*) *auto*
hence $(\lambda x. \psi x - x) \in \Theta(\lambda x. sum\text{-upto} (dirichlet\text{-prod} \mu f) x + (frac\ x + 2 * C))$
by (*simp only: landau-theta.uminus*)
also have $(\lambda x. sum\text{-upto} (dirichlet\text{-prod} \mu f) x + (frac\ x + 2 * C)) \in o(\lambda x. x)$
using $\langle sum\text{-upto} (dirichlet\text{-prod} \mu f) \in o(\lambda x. x) \rangle$ **by** (*rule sum-in-smallo*)
real-asymp+
finally show *?thesis* **by** (*rule smallo-imp-asymp-equiv*)
qed

We now turn to a related fact: For the weighted sum $A(x) := \sum_{n \leq x} \mu(n)/n$, the asymptotic relation $A(x) \in o(1)$ is also equivalent to the Prime Number Theorem. Like Apostol, we only show one direction, namely that $A(x) \in o(1)$ implies the PNT.

context

fixes A **defines** $A \equiv sum\text{-upto} (\lambda n. moebius\text{-mu} n / n)$
begin

lemma *sum-upto-moebius-mu-integral'*: $x > 1 \implies (A \text{ has-integral } x * A x - M x) \{1..x\}$

and *sum-upto-moebius-mu-integrable'*: $a \geq 1 \implies A \text{ integrable-on } \{a..b\}$

proof –

{
 fix $a\ b :: real$
 assume $ab: a \geq 1\ a < b$
 have $((\lambda t. A t * 1) \text{ has-integral } A b * b - A a * a - (\sum_{n \in real - \{a<..b\}. moebius\text{-mu} n / n * n}) \{a..b\})$
 unfolding *M-def A-def* **using** ab
 by (*intro partial-summation-strong [where X = {}]*)
 (*auto intro!: derivative-eq-intros continuous-intros simp flip: has-field-derivative-iff-has-vector-derivative*)
} **note** $* = this$
{
 fix $x :: real$ **assume** $x: x > 1$
 have [*simp*]: $A\ 1 = 1$ **by** (*simp add: A-def*)
 have $(\sum_{n \in real - \{1<..x\}. moebius\text{-mu} n / n * n} = (\sum_{n \in insert\ 1 (real - \{1<..x\}. moebius\text{-mu} n / n * n)} - 1)$
 using *finite-vimage-real-of-nat-greaterThanAtMost[of 1 x]* **by** (*subst sum.insert*)
auto
 also have $insert\ 1 (real - \{1<..x\}) = \{n. n > 0 \wedge real\ n \leq x\}$
 using x **by** *auto*
 also have $(\sum_{n \mid 0 < n \wedge real\ n \leq x. moebius\text{-mu} n / real\ n * real\ n} = M\ x$
 unfolding *M-def sum-upto-def* **by** (*intro sum.cong*) *auto*

```

    finally show (A has-integral x * A x - M x) {1..x} using *[of 1 x] x by (simp
add: mult-ac)
  }
  {
    fix a b :: real assume ab: a ≥ 1
    show A integrable-on {a..b}
    using *[of a b] ab
    by (cases a b rule: linorder-cases) (auto intro: integrable-negligible)
  }
qed

```

```

theorem sum-moebius-mu-div-n-smallo-imp-PNT:
  assumes smallo: A ∈ o(λ. 1)
  shows M ∈ o(λx. x) and ψ ~[at-top] (λx. x)
proof -
  have eventually (λx. M x = x * A x - integral {1..x} A) at-top
    using eventually-gt-at-top[of 1]
  by eventually-elim (use sum-upto-moebius-mu-integral' in ⟨simp add: has-integral-iff⟩)
  hence M ∈ Θ(λx. x * A x - integral {1..x} A)
    by (rule bigthetaI-cong)
  also have (λx. x * A x - integral {1..x} A) ∈ o(λx. x)
  proof (intro sum-in-smallo)
    from smallo show (λx. x * A x) ∈ o(λx. x)
    by (simp add: landau-divide-simps)
    show (λx. integral {1..x} A) ∈ o(λx. x)
    by (intro integral-smallo[OF smallo] sum-upto-moebius-mu-integrable')
      (auto intro!: derivative-eq-intros filterlim-ident)
  qed
  finally show M ∈ o(λx. x) .
  thus ψ ~[at-top] (λx. x)
    by (rule sum-moebius-mu-sublinear-imp-PNT)
qed

```

end

end

end

9 Elementary bounds on $\pi(x)$ and p_n

```

theory Elementary-Prime-Bounds
imports
  Prime-Number-Theorem.Prime-Counting-Functions
  Prime-Distribution-Elementary-Library
  More-Dirichlet-Misc
begin

```

In this section, we will follow Apostol and give elementary proofs of Chebyshev-type lower and upper bounds for $\pi(x)$, i.e. $c_1x/\ln x < \pi(x) < c_2x/\ln x$. From this, similar bounds for p_n follow as easy corollaries.

9.1 Preliminary lemmas

The following two estimates relating the central Binomial coefficient to powers of 2 and 4 form the starting point for Apostol's elementary bounds for $\pi(x)$:

lemma *twopow-le-central-binomial*: $2^n \leq \binom{2n}{n}$

proof –

have $2^n * \text{fact } n^2 \leq (\text{fact } (2 * n) :: \text{nat})$

proof (*induction n*)

case (*Suc n*)

have $(\text{fact } (2 * \text{Suc } n) :: \text{nat}) = (2 * n + 1) * (2 * n + 2) * \text{fact } (2 * n)$

by (*simp add: algebra-simps*)

have $2^{\text{Suc } n} * \text{fact } (\text{Suc } n)^2 = 2^n * \text{fact } n^2 * 2 * (n + 1)^2$

by (*simp add: algebra-simps power2-eq-square*)

also have $\dots \leq \text{fact } (2 * n) * 2 * (n + 1)^2$

by (*intro mult-right-mono Suc.IH*) *auto*

also have $\dots = \text{fact } (2 * n) * (2 * (n + 1))^2$

by (*simp add: mult-ac*)

also have $\dots \leq \text{fact } (2 * n) * ((2 * n + 1) * (2 * n + 2))$

by (*intro mult-left-mono*) (*auto simp: power2-eq-square*)

also have $\dots = \text{fact } (2 * \text{Suc } n)$

by (*simp add: algebra-simps*)

finally show *?case* .

qed *simp-all*

also have $\dots = \binom{2n}{n} * \text{fact } n^2$

using *binomial-fact-lemma*[of n $2 * n$] **by** (*simp add: power2-eq-square mult-ac*)

finally show *?thesis* **by** *simp*

qed

lemma *fourpow-gt-central-binomial*:

assumes $n > 0$

shows $4^n > \binom{2n}{n}$

proof –

have $(\sum_{i \in \{..2*n\} - \{n\}}. ((2 * n) \text{ choose } i)) > 0$

using *assms* **by** (*intro sum-pos*) (*auto simp: subset-iff*)

hence $(\binom{2n}{n}) < (\sum_{i \in \{..2*n\} - \{n\}}. ((2 * n) \text{ choose } i)) + ((2 * n) \text{ choose } n)$

by *simp*

also have $\dots = (\sum_{i \in \text{insert } n (\{..2*n\} - \{n\})}. ((2 * n) \text{ choose } i))$

by (*subst sum.insert*) *auto*

also have $\text{insert } n (\{..2*n\} - \{n\}) = \{..2*n\}$ **by** *auto*

also have $(\sum_{i \leq 2*n}. ((2 * n) \text{ choose } i)) = (1 + 1)^{(2 * n)}$

by (*subst binomial*) *simp-all*

also have $\dots = 4^{\wedge} n$
by (*subst power-mult*) (*simp add: eval-nat-numeral*)
finally show *?thesis* .
qed

9.2 Lower bound for $\pi(x)$

context

includes *prime-counting-notation*
fixes $S :: \text{nat} \Rightarrow \text{nat} \Rightarrow \text{int}$
defines $S \equiv (\lambda n p. (\sum m \in \{0 < .. \text{nat} \lfloor \log p (2 * n) \rfloor\}. \lfloor 2 * n / p^{\wedge} m \rfloor - 2 * \lfloor n / p^{\wedge} m \rfloor))$
begin

We now first prove the bound $\pi(x) \geq \frac{1}{6}x / \ln x$ for $x \geq 2$. The constant could probably be improved for starting points greater than 2; this is true for most of the constants in this section.

The first step is to show a slightly stronger bound for even numbers, where the constant is $\frac{1}{2} \ln 2 \approx 0.347$:

lemma

fixes $n :: \text{nat}$
assumes $n: n \geq 1$
shows $\pi\text{-bounds-aux}: \ln (\text{fact} (2 * n)) - 2 * \ln (\text{fact} n) =$
 $\text{prime-sum-upto} (\lambda p. S n p * \ln p) (2 * n)$
and $\pi\text{-lower-bound-ge-strong}: \pi (2 * n) \geq \ln 2 / 2 * (2 * n) / \ln (2 * n)$
proof –
define $L :: \text{real} \Rightarrow \text{nat} \Rightarrow \text{nat}$ **where** $L = (\lambda x p. \text{legendre-aux} x p)$
have $\ln (\text{fact} (2 * n)) - 2 * \ln (\text{fact} n) = \text{sum-upto} \ln (2 * n) - 2 * \text{sum-upto}$
 $\ln n$
by (*simp add: ln-fact-conv-sum-upto*)
also have $\dots = \text{prime-sum-upto} (\lambda p. L (2 * n) p * \ln p) (2 * n) -$
 $2 * \text{prime-sum-upto} (\lambda p. L n p * \ln p) n$
by (*subst (1 2) legendre-identity*) (*auto simp: L-def*)
also have $\text{prime-sum-upto} (\lambda p. L n p * \ln p) n = \text{prime-sum-upto} (\lambda p. L n p * \ln p) (2 * n)$
unfolding *prime-sum-upto-altdef2*
by (*intro sum.mono-neutral-left[OF finite-subset[of - {..2*n}]]*)
(auto dest: prime-gt-0-nat legendre-aux-posD
 $\text{simp: legendre-aux-eq-0 L-def le-nat-iff le-floor-iff}$)
also have $\text{prime-sum-upto} (\lambda p. L (2 * n) p * \ln p) (2 * n) -$
 $2 * \text{prime-sum-upto} (\lambda p. L n p * \ln p) (2 * n) =$
 $\text{prime-sum-upto} (\lambda p. (\text{real} (L (2 * n) p) - 2 * \text{real} (L n p)) * \ln p) (2$
 $* n)$
by (*simp add: ring-distrib sum-subtractf sum-distrib-left mult.assoc prime-sum-upto-def*)
also have $\dots = \text{prime-sum-upto} (\lambda p. \text{of-int} (S n p) * \ln p) (2 * n)$
unfolding *prime-sum-upto-def*
proof (*intro sum.cong refl, goal-cases*)
case (1 p)
define ub **where** $ub = \text{nat} \lfloor \log p (2 * n) \rfloor$
from 1 **have** $p: \text{prime } p \ p > 1 \ p \leq 2 * n$

using *prime-gt-1-nat*[of *p*] **by** *auto*
have $L (2 * n) p = (\sum m \in \{0 < .. ub\}. \text{nat } \lfloor 2 * n / p^m \rfloor)$
unfolding *L-def legendre-aux-altdef1* **using** *p 1* **by** (*auto simp: ub-def*)
moreover have $L n p = (\sum m \in \{0 < .. ub\}. \text{nat } \lfloor n / p^m \rfloor)$ **unfolding** *L-def*
proof (*intro legendre-aux-altdef2*)
have $\text{real } n = \text{real } p \text{ powr } \log p n$
using *n p* **by** *simp*
also have $\log (\text{real } p) 2 > 0$ **using** *p* **by** *auto*
hence $\log p n < 1 + \text{of-int } \lfloor \log p 2 + \log p n \rfloor$ **by** *linarith*
hence $\text{real } p \text{ powr } \log p n < \text{real } p \text{ powr } \text{Suc } ub$
unfolding *ub-def* **using** *n p* **by** (*intro powr-less-mono*) (*auto simp: log-mult*)
also have $\dots = p^{\text{Suc } ub}$
using *p* **by** (*subst powr-realpow*) *auto*
finally show $\text{real } n < \text{real } p^{\text{Suc } ub}$ **by** *simp*
qed (*use n p in auto*)
ultimately have $\text{real } (L (2 * n) p) - 2 * \text{real } (L n p) =$
 $(\sum m \in \{0 < .. ub\}. \text{real } (\text{nat } \lfloor 2 * n / p^m \rfloor) - 2 * \text{real } (\text{nat } \lfloor n / p^m \rfloor))$
by (*simp add: sum-subtractf sum-distrib-left*)
also have $\dots = \text{of-int } (\sum m \in \{0 < .. ub\}. \lfloor 2 * n / p^m \rfloor - 2 * \lfloor n / p^m \rfloor)$
unfolding *of-int-sum* **by** (*intro sum.cong*) *auto*
finally show *?case* **by** (*simp add: ub-def S-def*)
qed
finally show $\text{eq: } \ln (\text{fact } (2 * n)) - 2 * \ln (\text{fact } n) =$
 $\text{prime-sum-upto } (\lambda p. S n p * \ln p) (2 * n) .$

have *S-nonneg*: $S n p \geq 0$ **for** *p*
unfolding *S-def* **by** (*intro sum-nonneg*) *linarith*
have *S-le*: $S n p \leq \lfloor \log p (2 * n) \rfloor$ **if** *prime p* **for** *p*
proof –
have $S n p \leq (\sum m \in \{0 < .. \text{nat } \lfloor \log p (2 * n) \rfloor\}. 1)$
unfolding *S-def of-nat-mult of-nat-numeral* **by** (*intro sum-mono*) *linarith*
thus *?thesis* **using** *prime-gt-1-nat*[of *p*] **that** *n* **by** *auto*
qed

have $n * \ln 2 = \ln (\text{real } (2^n))$
by (*simp add: ln-realpow*)
also have $\dots \leq \ln (\text{real } ((2 * n) \text{ choose } n))$
using *twopow-le-central-binomial*[of *n*]
by (*subst ln-le-cancel-iff; (unfold of-nat-le-iff) ?*) *auto*
also have $\dots = \ln (\text{fact } (2 * n)) - 2 * \ln (\text{fact } n)$
by (*simp add: binomial-fact ln-div ln-mult*)
also have $\dots = \text{prime-sum-upto } (\lambda p. S n p * \ln p) (2 * n)$
by (*fact eq*)
also have $\dots \leq \text{prime-sum-upto } (\lambda p. \lfloor \log p (2 * n) \rfloor * \ln p) (2 * n)$
unfolding *prime-sum-upto-def* **using** *S-le*
by (*intro sum-mono mult-right-mono*) (*auto dest: prime-gt-0-nat*)
also have $\dots \leq \text{prime-sum-upto } (\lambda p. \ln (2 * n)) (2 * n)$
unfolding *prime-sum-upto-def*
proof (*intro sum-mono*)

```

fix p assume p ∈ {p. prime p ∧ real p ≤ real (2 * n)}
hence p: p > 1 using prime-gt-1-nat[of p] by auto
have real-of-int [log p (2 * n)] = real-of-int [ln (2 * n) / ln p]
  using p n by (simp add: log-def ln-mult)
also have ... ≤ ln (2 * n) / ln p
  by linarith
also have ... * ln p = ln (2 * n)
  using p by (simp add: field-simps)
finally show real-of-int [log p (2 * n)] * ln p ≤ ln (2 * n)
  using p by simp
qed
also have ... = ln (2 * n) * π (2 * n)
  by (simp add: π-def prime-sum-upto-def)
finally show π (2 * n) ≥ (ln 2 / 2) * (2 * n) / ln (2 * n)
  using n by (simp add: field-simps)
qed

lemma ln-2-ge-56-81: ln 2 ≥ (56 / 81 :: real)
  using ln-approx-bounds[of 2 2, simplified, simplified eval-nat-numeral, simplified]
  by simp

The bound for any real number  $x \geq 2$  follows fairly easily, although some
ugly accounting for error terms has to be done.

theorem π-lower-bound:
  fixes x :: real
  assumes x: x ≥ 2
  shows π x > (1 / 6) * (x / ln x)
proof (cases even (nat [x]))
case True
define n where n = nat [x]
from True assms have n: n ≥ 2 even n
  by (auto simp: n-def le-nat-iff le-floor-iff)
have (1 / 6) * (x / ln x) < (ln 2 / 4) * (x / ln x)
  using ln-2-ge-56-81 x by (intro mult-strict-right-mono) auto
also have ln 2 / 4 * (x / ln x) = (1 / 2) * (ln 2 / 2 * x / ln x)
  by simp
also have ... ≤ (1 - 1 / x) * (ln 2 / 2 * x / ln x)
  by (intro mult-right-mono) (use assms in ⟨auto simp: field-simps⟩)
also have (1 - 1 / x) * (ln 2 / 2 * x / ln x) = ln 2 / 2 * (x - 1) / ln x
  using assms by (simp add: field-simps)
also have ln 2 / 2 * (x - 1) / ln x ≤ ln 2 / 2 * n / ln n
  using x by (intro frac-le mult-mono mult-nonneg-nonneg) (auto simp: n-def)
also have ln 2 / 2 * n / ln n ≤ π n
  using π-lower-bound-ge-strong[of n div 2] ⟨even n⟩ n by simp
also have π n = π x by (simp add: n-def)
finally show ?thesis .
next
case False
define n where n = nat [x]

```

from *False assms* **have** $n: n \geq 2$ *odd* n
by (*auto simp: n-def le-nat-iff le-floor-iff*)
then obtain k **where** [*simp*]: $n = 2 * k + 1$
by (*auto elim!: oddE*)
from n **have** $k: k > 0$ **by** *simp*

from k **have** $3 \leq \text{real } n$ **by** *simp*
also have $\text{real } n \leq x$ **unfolding** *n-def* **using** x **by** *linarith*
finally have $x \geq 3$.

have $(1 / 6) * (x / \ln x) = 1 / 6 * x / \ln x$
using x **by** (*simp add: field-simps*)
also have $1 / 6 * x / \ln x < \ln 2 / 2 * (2 * k) / \ln (2 * k)$
proof (*intro frac-less*)
have $x < \text{real } n + 1$ **unfolding** *n-def* **by** *linarith*
hence $1 / 6 * x < 1 / 6 * (n + 1)$ **by** *simp*
also {
have $*$: $(3 * \ln 2 - 1 :: \text{real}) \geq 1$
using *ln-2-ge-56-81* **by** *simp*
hence $1 / (3 * \ln 2 - 1 :: \text{real}) \leq 1$ **by** *simp*
also have $1 \leq \text{real } k$ **using** k **by** *simp*
finally have $1 / 6 * (n + 1) \leq \ln 2 / 2 * \text{real } (2 * k)$
using $*$ **by** (*simp add: field-simps*)
}
finally show $1 / 6 * x < \ln 2 / 2 * \text{real } (2 * k)$.

next
have $\text{real } (2 * k) \leq \text{real } n$ **by** *simp*
also have $\dots \leq x$ **using** x **unfolding** *n-def* **by** *linarith*
finally show $\ln (\text{real } (2 * k)) \leq \ln x$ **using** k **by** *simp*
qed (*use k x in auto*)
also have $\ln 2 / 2 * (2 * k) / \ln (2 * k) \leq \pi (2 * k)$
by (*rule π -lower-bound-ge-strong*) (*use $\langle k > 0 \rangle$ in auto*)
also have $\pi (2 * k) \leq \pi n$
by (*rule π -mono*) *auto*
also have $\dots = \pi x$ **unfolding** *n-def* **by** *simp*
finally show *?thesis* .

qed

lemma *π -at-top: filterlim primes-pi at-top at-top*
proof (*rule filterlim-at-top-mono*)
show *eventually* $(\lambda x. \text{primes-pi } x \geq 1 / 6 * (x / \ln x))$ *at-top*
using *eventually-gt-at-top[of 2]* **by** *eventually-elim* (*intro less-imp-le π -lower-bound*)
qed *real-asymp*

9.3 Upper bound for $\vartheta(x)$

In this section, we prove a linear upper bound for ϑ . This is somewhat unnecessary because we already have a considerably better bound on $\vartheta(x)$ using a proof that has roughly the same complexity as this one and also

only uses elementary means. Nevertheless, here is the proof from Apostol's book; it is quite nice and it would be a shame not to formalise it.

The idea is to first show a bound for $\vartheta(2n) - \vartheta(n)$ and then deduce one for $\vartheta(2^n)$ from this by telescoping, which then yields one for general x by monotonicity.

lemma *ϑ -double-less*:

fixes $n :: nat$

assumes $n: n > 0$

shows $\vartheta (2 * real n) - \vartheta (real n) < real n * ln 4$

proof (*cases* $n \geq 2$)

case *False*

with *assms* **have** $n = 1$ **by** *force*

moreover **have** $\vartheta 2 = ln 2$ **by** (*simp add: eval- ϑ*)

ultimately show *?thesis* **by** *auto*

next

define P **where** $P = (\lambda n::nat. \{p \in \{0 <..n\}. prime\ p\})$

have *ϑ -eq*: $\vartheta n = (\sum p \in P n. ln p)$ **for** n

unfolding *ϑ -def prime-sum-upto-def*

by (*intro sum.cong*) (*auto simp: P-def dest: prime-gt-0-nat*)

have $\vartheta (2 * n) - \vartheta n = (\sum p \in P (2*n) - P n. ln p)$

unfolding *ϑ -eq* **by** (*rule Groups-Big.sum-diff [symmetric]*) (*auto simp: P-def*)

also have $(\sum p \in P (2*n) - P n. ln p) =$

$(\sum p \in P (2*n) - P n. (\lfloor 2*n/p \rfloor - 2 * \lfloor n/p \rfloor) * ln p)$

proof (*intro sum.cong refl*)

fix p **assume** $p: p \in P (2*n) - P n$

hence $*$: $real n / real p < 1$ $real n / real p \geq 1 / 2$ **by** (*auto simp: P-def*)

from $*$ **have** $\lfloor real n / real p \rfloor = 0$ **by** *linarith*

moreover from $*$ **have** $\lfloor 2 * real n / real p \rfloor = 1$

by *linarith*

ultimately show $ln p = (\lfloor 2*n/p \rfloor - 2 * \lfloor n/p \rfloor) * ln p$

by *simp*

qed

also have $(\sum p \in P (2*n) - P n. (\lfloor 2*n/p \rfloor - 2 * \lfloor n/p \rfloor) * ln p) \leq$

$(\sum p \in P (2*n). (\lfloor 2*n/p \rfloor - 2 * \lfloor n/p \rfloor) * ln p)$

proof (*intro sum-mono2*)

fix p **assume** $p: p \in P (2 * n) - (P (2 * n) - P n)$

have $2 * \lfloor real n / real p \rfloor \leq \lfloor 2 * (real n / real p) \rfloor$

by *linarith*

thus $0 \leq real-of-int (\lfloor real (2 * n) / real p \rfloor - 2 * \lfloor real n / real p \rfloor) * ln (real p)$

using p **by** (*intro mult-nonneg-nonneg*) (*auto simp: P-def*)

qed (*auto simp: P-def*)

also have $*$: $2 * real-of-int \lfloor real n / real p \rfloor^m \leq 2 * real n / real p \wedge^m$ **for**
 $p \ m$

by *linarith*
have $(\sum m \in \{1\}. \lfloor 2 * \text{real } n / \text{real } p^{\wedge} m \rfloor - 2 * \lfloor n / \text{real } p^{\wedge} m \rfloor) \leq S n p$
if *prime* $p \leq 2 * n$ **for** $p :: \text{nat}$
unfolding *S-def* **using** *prime-gt-1-nat* [*OF that(1)*] *that(2)* n *[of p]*
by (*intro sum-mono2*) (*auto dest: prime-gt-1-nat simp: le-nat-iff le-floor-iff*)
hence $(\sum p \in P (2*n). (\sum m \in \{1\}. \lfloor 2*n/p^{\wedge} m \rfloor - 2 * \lfloor n/p^{\wedge} m \rfloor) * \ln p) \leq (\sum p \in P (2*n). S n p * \ln p)$
by (*intro sum-mono mult-right-mono; (unfold of-int-le-iff)?*)
(auto dest: prime-gt-1-nat simp: P-def)
hence $(\sum p \in P (2*n). (\lfloor 2*n/p \rfloor - 2 * \lfloor n/p \rfloor) * \ln p) \leq (\sum p \in P (2*n). S n p * \ln p)$
by *simp*

also have $(\sum p \in P (2*n). S n p * \ln p) = \text{prime-sum-upto } (\lambda p. S n p * \ln p) (2 * n)$
unfolding *P-def prime-sum-upto-def* **by** (*intro sum.cong*) (*auto simp: P-def dest: prime-gt-0-nat*)
also have $\dots = \ln (\text{fact } (2 * n)) - 2 * \ln (\text{fact } n)$
by (*rule π -bounds-aux [symmetric]*) (*use n in auto*)
also have $\dots = \ln (\text{real } ((2*n) \text{ choose } n))$
by (*simp add: binomial-fact ln-div ln-mult*)
also have $\dots < \ln (\text{real } (4^{\wedge} n))$
by (*subst ln-less-cancel-iff; (unfold of-nat-le-iff)?*)
(use fourpow-gt-central-binomial[of n] n in auto)
also have $\dots = n * \ln 4$
by (*simp add: ln-realpow*)
finally show *?thesis* **by** *simp*
qed

lemma *ϑ -twopow-less*: $\vartheta (2^{\wedge} r) < 2^{\wedge} (r + 1) * \ln 2$

proof –

have *ϑ -twopow-diff*: $\vartheta (2^{\wedge} \text{Suc } k) - \vartheta (2^{\wedge} k) < 2^{\wedge} \text{Suc } k * \ln 2$ **for** k

using *ϑ -double-less* [of $2^{\wedge} k$] *ln-realpow* [of 2] **by** *simp*

show $\vartheta (2^{\wedge} r) < 2^{\wedge} (r + 1) * \ln 2$

proof (*cases r > 0*)

case *True*

have $(\sum k < r. \vartheta (2^{\wedge} \text{Suc } k) - \vartheta (2^{\wedge} k)) < (\sum k < r. 2^{\wedge} \text{Suc } k * \ln 2)$

by (*intro sum-strict-mono ϑ -twopow-diff*) (*use <r > 0 in auto*)

also have $(\sum k < r. \vartheta (2^{\wedge} \text{Suc } k) - \vartheta (2^{\wedge} k)) = \vartheta (2^{\wedge} r)$

by (*subst sum-lessThan-telescope*) *auto*

also have $(\sum k < r. 2^{\wedge} \text{Suc } k * \ln 2 :: \text{real}) = (\sum k < r. 2^{\wedge} k) * 2 * \ln 2$

by (*simp add: sum-distrib-right sum-distrib-left mult-ac*)

also have $(\sum k < r. 2^{\wedge} k :: \text{real}) = 2^{\wedge} r - 1$

using *geometric-sum* [of $2 :: \text{real}$] **by** *simp*

also have $\dots \leq 2^{\wedge} r$ **by** *simp*

finally show $\vartheta (2^{\wedge} r) < 2^{\wedge} (r + 1) * \ln 2$

by *simp*

qed *auto*

qed

```

theorem  $\vartheta$ -upper-bound-weak:
  fixes  $n :: \text{nat}$ 
  assumes  $n: n > 0$ 
  shows  $\vartheta n < 4 * \ln 2 * n$ 
proof -
  define  $r$  where  $r = \text{Discrete.log } n$ 
  have  $\vartheta n \leq \vartheta (\text{real } (2 \wedge \text{Suc } r))$ 
  unfolding  $r$ -def using  $\text{log-exp2-ge}$ [of  $n$ ] by ( $\text{intro } \vartheta$ -mono,  $\text{unfold of-nat-le-iff}$ )
  auto
  also have  $\dots < 4 * \ln 2 * \text{real } (2 \wedge r)$ 
  using  $\vartheta$ -twopow-less[of  $r + 1$ ] by ( $\text{simp add: mult-ac}$ )
  also have  $\dots \leq 4 * \ln 2 * \text{real } n$  unfolding  $r$ -def
  by ( $\text{intro mult-left-mono}$ ,  $\text{unfold of-nat-le-iff}$ ,  $\text{intro log-exp2-le}$ ) ( $\text{use } n$  in  $\text{auto}$ )
  finally show  $\vartheta n < 4 * \ln 2 * n$  by  $\text{simp}$ 
qed

```

9.4 Upper bound for $\pi(x)$

We use our upper bound for $\vartheta(x)$ (the strong one, not the one from the previous section) to derive an upper bound for $\pi(x)$.

As a first step, we show the following lemma about the global maximum of the function $\ln x/x^c$ for $c > 0$:

```

lemma  $\pi$ -upper-bound-aux:
  fixes  $c :: \text{real}$ 
  assumes  $c > 0$ 
  defines  $f \equiv (\lambda x. x \text{ powr } (-c) * \ln x)$ 
  assumes  $x: x > 0$ 
  shows  $f x \leq 1 / (c * \exp 1)$ 
proof -
  define  $f'$  where  $f' = (\lambda x. x \text{ powr } (-c - 1) * (1 - c * \ln x))$ 
  define  $z$  where  $z = \exp (1 / c)$ 
  have  $z > 0$  by ( $\text{simp add: z-def}$ )
  have  $\text{deriv: } (f \text{ has-real-derivative } f' t) \text{ (at } t) \text{ if } t > 0$  for  $t$ 
  unfolding  $f$ -def  $f'$ -def using  $\text{that}$ 
  by ( $\text{auto intro!: derivative-eq-intros simp: field-simps powr-diff powr-minus}$ )
  have [ $\text{simp}$ ]:  $f z = 1 / (c * \exp 1)$ 
  by ( $\text{simp add: z-def f-def powr-def exp-minus field-simps}$ )

  show  $?thesis$ 
proof ( $\text{cases } x z \text{ rule: linorder-cases}$ )
  assume  $x: x < z$ 
  from  $x$   $\text{assms}$  have  $t: \exists t. t > x \wedge t < z \wedge f z - f x = (z - x) * f' t$ 
  by ( $\text{intro MVT2 deriv}$ )  $\text{auto}$ 
  then obtain  $t$  where  $t: t > x \wedge t < z \wedge f z - f x = (z - x) * f' t$ 
  by  $\text{blast}$ 
  hence  $\ln t < \ln z$  using  $\text{assms}$  by  $\text{simp}$ 
  also have  $\ln z = 1 / c$  by ( $\text{simp add: z-def}$ )

```

```

finally have  $0 \leq (z - x) * f' t$ 
  unfolding f'-def using  $\langle c > 0 \rangle x$ 
  by (intro mult-nonneg-nonneg) (auto simp: z-def field-simps)
also from  $t$  have  $\dots = f z - f x$  by (simp add: algebra-simps)
finally show ?thesis by simp
next
assume  $x: x > z$ 
from  $x$  assms  $\langle z > 0 \rangle$  have  $t: \exists t. t > z \wedge t < x \wedge f x - f z = (x - z) * f' t$ 
  by (intro MVT2 deriv) auto
then obtain  $t$  where  $t: t > z \wedge t < x \wedge f x - f z = (x - z) * f' t$ 
  by blast
hence  $\ln z < \ln t$  using  $\langle z > 0 \rangle$  assms by simp
also have  $\ln z = 1 / c$  by (simp add: z-def)
finally have  $0 \leq (z - x) * f' t$ 
  unfolding f'-def using  $\langle c > 0 \rangle x$ 
  by (intro mult-nonpos-nonpos mult-nonneg-nonpos) (auto simp: z-def field-simps)
also from  $t$  have  $\dots = f z - f x$  by (simp add: algebra-simps)
finally show ?thesis by simp
qed auto
qed

```

Following Apostol, we first show a generic bound depending on some real-valued parameter α :

```

lemma  $\pi$ -upper-bound-strong:
  fixes  $\alpha :: \text{real}$  and  $n :: \text{nat}$ 
  assumes  $n: n \geq 2$  and  $\alpha: \alpha \in \{0 < .. < 1\}$ 
  shows  $\pi n < (1 / ((1 - \alpha) * \exp 1) + \ln 4 / \alpha) * n / \ln n$ 
proof -
  have real n powr  $\alpha \leq$  real n powr 1
    using assms n by (intro powr-mono) auto
  hence  $n'$ : real n powr  $\alpha \leq$  real n using  $n$  by simp

  define  $P$  where  $P = (\lambda x. \{p. \text{prime } p \wedge \text{real } p \leq x\})$ 
  define  $Q$  where  $Q = \{p. \text{prime } p \wedge \text{real } p \in \{n \text{ powr } \alpha < .. n\}\}$ 

  have finite-P [intro]: finite (P x) for  $x$ 
proof (cases x  $\geq$  0)
  case True
  hence  $P x \subseteq \{.. \text{nat } \lfloor x \rfloor\}$ 
    by (auto simp: le-nat-iff le-floor-iff P-def)
  thus ?thesis by (rule finite-subset) auto
qed (auto simp: P-def)

  have P-subset: P x  $\subseteq$  P y if x  $\leq$  y for  $x y$ 
    using that by (auto simp: P-def)

  have  $Q = P n - P (n \text{ powr } \alpha)$  by (auto simp: Q-def P-def)
  also have  $\text{card } \dots = \text{card } (P n) - \text{card } (P (n \text{ powr } \alpha))$ 
    by (intro card-Diff-subset finite-P P-subset n')

```

also have $real \dots = \pi n - \pi (n \text{ powr } \alpha)$
by (*subst of-nat-diff*[*OF card-mono*[*OF - P-subset*]])
(use n' in $\langle auto simp: \pi\text{-def prime-sum-upto-def P-def \rangle$)
finally have $card\text{-}Q: real (card Q) = \pi n - \pi (n \text{ powr } \alpha)$.

have $(\pi n - \pi (n \text{ powr } \alpha)) * ln (n \text{ powr } \alpha) = (\sum p \in Q. ln (n \text{ powr } \alpha))$
using *card-Q by simp*
also have $\dots \leq (\sum p \in Q. ln p)$
using $n \alpha$ **by** (*intro sum-mono, subst ln-le-cancel-iff*) (*auto simp: Q-def dest: prime-gt-0-nat*)
also have $\dots \leq \vartheta n$
unfolding $\vartheta\text{-def prime-sum-upto-def}$ **by** (*intro sum-mono2*) (*auto simp: Q-def dest: prime-gt-1-nat*)
also have $\dots < ln 4 * real n$
by (*rule ϑ -upper-bound*) (*use n in auto*)
finally have *ineq*: $(\pi n - \pi (n \text{ powr } \alpha)) * ln (n \text{ powr } \alpha) < ln 4 * n$.

with n assms have $\pi n < \pi (n \text{ powr } \alpha) + (ln 4 / \alpha) * n / ln n$
by (*simp add: field-simps ln-powr*
del: div-mult-self3 div-mult-self4 div-mult-self2 div-mult-self1)
also have $\pi (n \text{ powr } \alpha) \leq n \text{ powr } \alpha$
by (*rule π -le-self*) *auto*
also have $n \text{ powr } \alpha + ln 4 / \alpha * n / ln n =$
 $(n \text{ powr } (-(1 - \alpha)) * ln n + ln 4 / \alpha) * n / ln n$
using $n \alpha$ **by** (*simp add: field-simps powr-diff*
del: div-mult-self3 div-mult-self4 div-mult-self2 div-mult-self1)
also have $n \text{ powr } (-(1 - \alpha)) * ln n \leq 1 / ((1 - \alpha) * exp 1)$
by (*intro π -upper-bound-aux*) (*use αn in auto*)
hence $(n \text{ powr } (-(1 - \alpha)) * ln n + ln 4 / \alpha) * n / ln n \leq$
 $(1 / ((1 - \alpha) * exp 1) + ln 4 / \alpha) * n / ln n$
using $n \alpha$ **by** (*intro divide-right-mono mult-right-mono add-mono*) *auto*
finally show $\pi n < (1 / ((1 - \alpha) * exp 1) + ln 4 / \alpha) * n / ln n$
by *simp*

qed

The choice $\alpha := \frac{2}{3}$ then leads to the upper bound $\pi(x) < cx / \ln x$ with $c = 3(e^{-1} + \ln 2) \approx 3.183$. This is considerably stronger than Apostol's bound.

theorem *π -upper-bound*:

fixes $x :: real$

assumes $x \geq 2$

shows $\pi x < 3 * (exp (-1) + ln 2) * x / ln x$

proof (*cases $x \geq 3$*)

case *False*

have $\pi x = \pi (nat \lfloor x \rfloor)$ **by** *simp*

also from *False* **and** *assms* **have** $nat \lfloor x \rfloor = 2$

by *linarith*

finally have $\pi x = 1$ **by** (*simp add: eval- π*)

also have $\dots < 3 * (exp (-1) + ln 2) * exp 1$

```

    by (simp add: exp-minus field-simps add-pos-pos
        del: div-mult-self3 div-mult-self4 div-mult-self2 div-mult-self1)
  also have ... ≤ 3 * (exp (-1) + ln 2) * (x / ln x)
    using π-upper-bound-aux[of 1 x]
    by (intro mult-left-mono) (use assms in ⟨auto simp: field-simps powr-minus⟩)
  finally show ?thesis
    using assms by (simp add: field-simps)
next
case True
define n where n = nat ⌊x⌋
from True have n: n ≥ 3 by (simp add: n-def le-nat-iff le-floor-iff)
have π x = π n
  by (simp add: n-def)
also have π n < 3 * (exp (-1) + ln 2) * (n / ln n)
  using π-upper-bound-strong[of n 2 / 3] ln-realpow[of 2 2] n
  by (simp add: field-simps exp-minus
      del: div-mult-self3 div-mult-self4 div-mult-self2 div-mult-self1)
also have ... ≤ 3 * (exp (-1) + ln 2) * (x / ln x)
  using n True by (intro mult-left-mono divide-ln-mono) (auto simp: n-def)
  finally show ?thesis by (simp add: divide-simps)
qed

corollary π-upper-bound':
  fixes x :: real
  assumes x ≥ 2
  shows π x < 443 / 139 * (x / ln x)
proof -
  have 2.71828 ≤ 5837465777 / 2147483648 - inverse (2 ^ 32 :: real)
    by simp
  also have ... ≤ exp (1 :: real)
    using e-approx-32 by linarith
  finally have exp 1 ≥ (2.71828 :: real) .
  hence e-m1: exp (-1) ≤ (10^5 / 271828 :: real) by (simp add: field-simps
  exp-minus)

  from assms have π x < 3 * (exp (-1) + ln 2) * (x / ln x)
    using π-upper-bound[of x] by (simp add: field-simps)
  also have ... ≤ 443 / 139 * (x / ln x)
  proof (intro mult-right-mono)
    have 3 * (exp (-1) + ln 2 :: real) ≤ 3 * (10^5 / 271828 + 25 / 36)
      using e-m1 by (intro mult-left-mono add-mono ln2-le-25-over-36)
      (auto simp: exp-minus field-simps abs-if split: if-splits)
    also have ... ≤ 443 / 139 by simp
    finally show 3 * (exp (-1) + ln 2 :: real) ≤ 443 / 139 by simp
  qed (use assms in auto)
  finally show ?thesis .
qed

corollary π-upper-bound'':

```

```

fixes  $x :: \text{real}$ 
assumes  $x \geq 2$ 
shows  $\pi x < 4 * (x / \ln x)$ 
by (rule less-le-trans[OF  $\pi$ -upper-bound'[OF assms] mult-right-mono]) (use assms
in auto)

```

In particular, we have now shown a weak version of the Prime Number Theorem, namely that $\pi(x) \in \Theta(x / \ln x)$:

lemma π -bigtheta: $\pi \in \Theta(\lambda x. x / \ln x)$

proof

```

have eventually ( $\lambda x. |\pi x| \leq 3 * (\exp (- 1) + \ln 2) * |x / \ln x|$ ) at-top
  using eventually-ge-at-top[of 2]
  by eventually-elim (use  $\pi$ -upper-bound in  $\langle \text{auto intro!; less-imp-le} \rangle$ )
thus  $\pi \in O(\lambda x. x / \ln x)$ 
  by (intro bigO[where  $c = 3 * (\exp (- 1) + \ln 2)$ ]) auto
next
have eventually ( $\lambda x. |\pi x| \geq 1 / 6 * |x / \ln x|$ ) at-top
  using eventually-ge-at-top[of 2]
  by eventually-elim (use  $\pi$ -lower-bound in  $\langle \text{auto intro!; less-imp-le} \rangle$ )
thus  $\pi \in \Omega(\lambda x. x / \ln x)$ 
  by (intro landau-omega.bigI[where  $c = 1 / 6$ ]) auto
qed

```

9.5 Bounds for p_n

By some rearrangements, the lower and upper bounds for $\pi(x)$ give rise to analogous bounds for p_n :

lemma *nth-prime-lower-bound-gen*:

```

assumes  $c : c > 0$  and  $n : n > 0$ 
assumes  $\bigwedge n. n \geq 2 \implies \pi (\text{real } n) < (1 / c) * (\text{real } n / \ln (\text{real } n))$ 
shows nth-prime  $(n - 1) \geq c * (\text{real } n * \ln (\text{real } n))$ 

```

proof –

```

define  $p$  where  $p = \text{nth-prime } (n - 1)$ 
have  $p \geq 2$ 
  by (simp add: p-def nth-prime-ge-2)
have  $p \geq n$  using nth-prime-lower-bound[of  $n - 1$ ] by (simp add: p-def)

have  $c * (n * \ln n) \leq c * (n * \ln p)$ 
  using  $n < p \geq n$  by (intro mult-left-mono auto)
also {
  from  $\langle p \geq 2 \rangle$  have  $\pi (\text{real } p) < (1 / c) * (\text{real } p / \ln (\text{real } p))$ 
    by (rule assms)
  also from  $n$  have  $\pi (\text{real } p) = n$ 
    by (simp add: p-def)
  finally have  $c * (n * \ln p) < p$ 
    using  $c < p \geq 2 \rangle n$  by (simp add: field-simps)
}
finally show nth-prime  $(n - 1) \geq c * (\text{real } n * \ln (\text{real } n))$ 

```

using $c\ n$ by (*simp add: p-def*)
qed

corollary *nth-prime-lower-bound*:

$n > 0 \implies \text{nth-prime } (n - 1) \geq (139 / 443) * (n * \ln n)$
 using π -upper-bound' by (*intro nth-prime-lower-bound-gen*) *auto*

corollary *nth-prime-upper-bound*:

assumes $n: n > 0$
 shows $\text{nth-prime } (n - 1) < 12 * (n * \ln n + n * \ln (12 / \exp 1))$

proof –

define p where $p = \text{nth-prime } (n - 1)$
 have $p \geq 2$
 by (*simp add: p-def nth-prime-ge-2*)

have $(1 / 6) * (p / \ln p) < \pi\ p$
 by (*intro π -lower-bound*) (*use $\langle p \geq 2 \rangle$ in auto*)

also have $\dots = n$

using n by (*simp add: p-def*)
 finally have *less*: $p < 6 * n * \ln p$
 using $\langle p \geq 2 \rangle$ by (*simp add: field-simps*)

also have $\ln p \leq (2 / \exp 1) * \text{sqrt } p$
 using π -upper-bound-aux[*of 1 / 2 p*] $\langle p \geq 2 \rangle$
 by (*simp add: field-simps powr-minus powr-half-sqrt*)
 finally have $\text{sqrt } p * \text{sqrt } p < 12 / \exp 1 * n * \text{sqrt } p$
 using n by *simp*

hence $\text{sqrt } p < 12 / \exp 1 * n$
 by (*subst (asm) mult-less-cancel-right*) (*use $\langle p \geq 2 \rangle$ in auto*)

hence $\ln (\text{sqrt } p) < \ln (12 / \exp 1 * n)$
 using $n \langle p \geq 2 \rangle$ by (*subst ln-less-cancel-iff*) *auto*

also have $\ln (\text{sqrt } p) = \ln p / 2$
 using $\langle p \geq 2 \rangle$ by (*simp add: ln-sqrt*)

also have $\ln (12 / \exp 1 * n) = \ln n + \ln (12 / \exp 1)$
 using n by (*simp add: ln-div ln-mult*)

finally have *ln-less*: $\ln p \leq 2 * \ln n + 2 * \ln (12 / \exp 1)$
 by *simp*

have $p < 6 * n * \ln p$ by (*fact less*)

also have $\dots \leq 6 * n * (2 * \ln n + 2 * \ln (12 / \exp 1))$
 by (*intro mult-left-mono ln-less*) *auto*

also have $\dots = 12 * (n * \ln n + n * \ln (12 / \exp 1))$
 by (*simp add: algebra-simps*)

finally show *?thesis* unfolding *p-def* .

qed

We can thus also conclude that $p_n \sim n \ln n$:

corollary *nth-prime-bigtheta*: $\text{nth-prime} \in \Theta(\lambda n. n * \ln n)$

proof


```

have eventually ( $\lambda n. |nth\text{-prime } n| \leq$ 
   $12 * |(n + 1) * \ln (n + 1) + (n + 1) * \ln (12 / \exp 1)|$ ) at-top
  using eventually-ge-at-top[of 2]
proof eventually-elim
  case (elim n)
  with nth-prime-upper-bound[of n + 1] show ?case by (auto simp: add-ac)
qed
hence nth-prime  $\in O(\lambda n. (n + 1) * \ln (n + 1) + (n + 1) * \ln (12 / \exp 1))$ 
  by (intro bigO[where c = 12]) auto
also have ( $\lambda n. (n + 1) * \ln (n + 1) + (n + 1) * \ln (12 / \exp 1)$ )  $\in O(\lambda n::nat.$ 
 $n * \ln n)$ 
  by real-asymp
finally show nth-prime  $\in O(\lambda n. n * \ln n)$  .
next
have eventually ( $\lambda n. |nth\text{-prime } n| \geq 139 / 443 * |(n + 1) * \ln (n + 1)|$ ) at-top
  using eventually-ge-at-top[of 2]
proof eventually-elim
  case (elim n)
  with nth-prime-lower-bound[of n + 1] show ?case by (auto simp: add-ac)
qed
hence nth-prime  $\in \Omega(\lambda n::nat. \text{real } (n + 1) * \ln (\text{real } n + 1))$ 
  by (intro landau-omega.bigO[where c = 139 / 443]) (auto simp: add-ac)
also have ( $\lambda n::nat. \text{real } (n + 1) * \ln (\text{real } n + 1)$ )  $\in \Omega(\lambda n. n * \ln n)$ 
  by real-asymp
finally show nth-prime  $\in \Omega(\lambda n. n * \ln n)$  .
qed

end

end

```

10 The asymptotics of the summatory divisor σ function

```

theory Summatory-Divisor-Sigma-Bounds
  imports Partial-Zeta-Bounds More-Dirichlet-Misc
begin

```

In this section, we analyse the asymptotic behaviour of the summatory divisor functions $\sum_{n \leq x} \sigma_\alpha(n)$ for real α . This essentially tells us what the average value of these functions is for large x .

The case $\alpha = 0$ is not treated here since σ_0 is simply the divisor function, for which precise asymptotics are already available in the AFP.

10.1 Case 1: $\alpha = 1$

If $\alpha = 1$, $\sigma_\alpha(n)$ is simply the sum of all divisors of n . Here, the asymptotics is

$$\sum_{n \leq x} \sigma_1(n) = \frac{\pi^2}{12} x^2 + O(x \ln x) .$$

theorem *summatory-divisor-sum-asymptotics:*

*sum-upto divisor-sum = o ($\lambda x. \pi^2 / 12 * x^2$) + o ($O(\lambda x. x * \ln x)$)*

proof –

define ζ **where** $\zeta = \text{Re } (\text{zeta } 2)$

define $R1$ **where** $R1 = (\lambda x. \text{sum-upto real } x - x^2 / 2)$

define $R2$ **where** $R2 = (\lambda x. \text{sum-upto } (\lambda d. 1 / d^2) x - (\zeta - 1 / x))$

obtain $c1$ **where** $c1: c1 > 0 \wedge x. x \geq 1 \implies |R1 x| \leq c1 * x$

using *zeta-partial-sum-le-neg[of 1]* **by** (*auto simp: R1-def*)

obtain $c2$ **where** $c2: c2 > 0 \wedge x. x \geq 1 \implies |R2 x| \leq c2 / x^2$

using *zeta-partial-sum-le-pos[of 2]*

by (*auto simp: ζ -def R2-def powr-minus field-simps*)

simp del: div-mult-self3 div-mult-self4 div-mult-self2 div-mult-self1)

have $le: |\text{sum-upto divisor-sum } x - \zeta / 2 * x^2| \leq c2 / 2 + x / 2 + c1 * x * (\ln x + 1)$

if $x: x \geq 1$ **for** x

proof –

have $div-le: \text{real } (a \text{ div } b) \leq x$ **if** $a \leq x$ **for** $a b :: \text{nat}$

by (*rule order.trans[OF - that(1)]*) *auto*

have $\text{real } (\text{sum-upto divisor-sum } x) = \text{sum-upto } (\text{dirichlet-prod real } (\lambda-. 1)) x$

by (*simp add: divisor-sigma-conv-dirichlet-prod [abs-def]*)

sum-upto-def divisor-sigma-1-left [symmetric])

also have $\dots = \text{sum-upto } (\lambda n. \sum d \mid d \text{ dvd } n. \text{real } d) x$

by (*simp add: dirichlet-prod-def*)

also have $\dots = (\sum (n, d) \in (\text{SIGMA } n: \{n. n > 0 \wedge \text{real } n \leq x\}. \{d. d \text{ dvd } n\}). \text{real } d)$

unfolding *sum-upto-def* **by** (*subst sum.Sigma*) *auto*

also have $\dots = (\sum (d, q) \in (\text{SIGMA } d: \{d. d > 0 \wedge \text{real } d \leq x\}. \{q. q > 0 \wedge \text{real } q \leq x / d\}). \text{real } q)$

by (*rule sum.reindex-bij-witness[of - $\lambda(d, q). (d * q, q) \lambda(n, d). (n \text{ div } d, d)$]*)

(use div-le in <auto simp: field-simps>)

also have $\dots = \text{sum-upto } (\lambda d. \text{sum-upto real } (x / d)) x$

by (*simp add: sum-upto-def sum.Sigma*)

also have $\dots = x^2 * \text{sum-upto } (\lambda d. 1 / d^2) x / 2 + \text{sum-upto } (\lambda d. R1 (x / d)) x$

by (*simp add: R1-def sum-upto-def sum.distrib sum-subtractf sum-divide-distrib*)

power-divide sum-distrib-left)

also have $\text{sum-upto } (\lambda d. 1 / d^2) x = \zeta - 1 / x + R2 x$

by (*simp add: R2-def*)

finally have $eq: \text{real } (\text{sum-upto divisor-sum } x) =$

$x^2 * (\zeta - 1 / x + R2 x) / 2 + \text{sum-upto } (\lambda d. R1 (x / \text{real } d))$

x .

```

have real (sum-upto divisor-sum x) - ζ / 2 * x2 =
  x2 / 2 * R2 x - x / 2 + sum-upto (λd. R1 (x / real d)) x using x
by (subst eq)
  (simp add: field-simps power2-eq-square del: div-diff div-add
  del: div-mult-self3 div-mult-self4 div-mult-self2 div-mult-self1)
also have |...| ≤ c2 / 2 + x / 2 + c1 * x * (ln x + 1)
proof (rule order.trans[OF abs-triangle-ineq] order.trans[OF abs-triangle-ineq4]
add-mono)+
  have |x2 / 2 * R2 x| = x2 / 2 * |R2 x|
  using x by (simp add: abs-mult)
  also have ... ≤ x2 / 2 * (c2 / x2)
  using x by (intro mult-left-mono c2) auto
  finally show |x2 / 2 * R2 x| ≤ c2 / 2
  using x by simp
next
have |sum-upto (λd. R1 (x / real d)) x| ≤ sum-upto (λd. |R1 (x / real d)|) x
  unfolding sum-upto-def by (rule sum-abs)
  also have ... ≤ sum-upto (λd. c1 * (x / real d)) x
  unfolding sum-upto-def by (intro sum-mono c1) auto
  also have ... = c1 * x * sum-upto (λd. 1 / real d) x
  by (simp add: sum-upto-def sum-distrib-left)
  also have sum-upto (λd. 1 / real d) x = harm (nat ⌊x⌋)
  unfolding sum-upto-altdef harm-def by (intro sum.cong) (auto simp:
field-simps)
  also have ... ≤ ln (nat ⌊x⌋) + 1
  by (rule harm-le) (use x in ⟨auto simp: le-nat-iff⟩)
  also have ln (nat ⌊x⌋) ≤ ln x using x by simp
  finally show |sum-upto (λd. R1 (x / real d)) x| ≤ c1 * x * (ln x + 1)
  using c1(1) x by simp
qed (use x in auto)
finally show |sum-upto divisor-sum x - ζ / 2 * x2| ≤ c2 / 2 + x / 2 + c1
* x * (ln x + 1) .
qed

have eventually (λx. |sum-upto divisor-sum x - ζ / 2 * x2| ≤
  c2 / 2 + x / 2 + c1 * x * (ln x + 1)) at-top
using eventually-ge-at-top[of 1] by eventually-elim (use le in auto)
hence eventually (λx. |sum-upto divisor-sum x - ζ / 2 * x2| ≤
  |c2 / 2 + x / 2 + c1 * x * (ln x + 1)|) at-top
by eventually-elim linarith
hence (λx. sum-upto divisor-sum x - ζ / 2 * x2) ∈ O(λx. c2 / 2 + x / 2 + c1
* x * (ln x + 1))
by (intro landau-o.bigI[of 1]) auto
also have (λx. c2 / 2 + x / 2 + c1 * x * (ln x + 1)) ∈ O(λx. x * ln x)
by real-asymp
finally show ?thesis
by (subst set-minus-plus [symmetric])

```

(simp-all add: fun-diff-def algebra-simps ζ -def zeta-even-numeral)

qed

10.2 Case 2: $\alpha > 0, \alpha \neq 1$

Next, we consider the case $\alpha > 0$ and $\alpha \neq 1$. We then have:

$$\sum_{n \leq x} \sigma_\alpha(n) = \frac{\zeta(\alpha + 1)}{\alpha + 1} x^{\alpha+1} + O\left(x^{\max(1, \alpha)}\right)$$

theorem *summatory-divisor-sigma-asymptotics-pos*:

fixes $\alpha :: \text{real}$

assumes $\alpha: \alpha > 0 \ \alpha \neq 1$

defines $\zeta \equiv \text{Re } (\text{zeta } (\alpha + 1))$

shows *sum-upto (divisor-sigma α) = o*

*($\lambda x. \zeta / (\alpha + 1) * x \text{ powr } (\alpha + 1)$) + o ($\lambda x. x \text{ powr } \max 1 \ \alpha$)*

proof –

define *R1* **where** *R1* = ($\lambda x. \text{sum-upto } (\lambda d. \text{real } d \text{ powr } \alpha) \ x - x \text{ powr } (\alpha + 1) / (\alpha + 1)$)

define *R2* **where** *R2* = ($\lambda x. \text{sum-upto } (\lambda d. d \text{ powr } (-\alpha - 1)) \ x - (\zeta - x \text{ powr } -\alpha / \alpha)$)

define *R3* **where** *R3* = ($\lambda x. \text{sum-upto } (\lambda d. d \text{ powr } -\alpha) \ x - x \text{ powr } (1 - \alpha) / (1 - \alpha)$)

obtain *c1* **where** *c1*: $c1 > 0 \ \wedge x. x \geq 1 \implies |R1 \ x| \leq c1 * x \text{ powr } \alpha$

using *zeta-partial-sum-le-neg*[of α] α **by** (*auto simp: R1-def add-ac*)

obtain *c2* **where** *c2*: $c2 > 0 \ \wedge x. x \geq 1 \implies |R2 \ x| \leq c2 * x \text{ powr } (-\alpha - 1)$

using *zeta-partial-sum-le-pos*[of $\alpha + 1$] α **by** (*auto simp: ζ -def R2-def*)

obtain *c3* **where** *c3*: $c3 > 0 \ \wedge x. x \geq 1 \implies |R3 \ x| \leq c3$

using *zeta-partial-sum-le-pos*'[of α] α **by** (*auto simp: R3-def*)

define *ub* :: *real* \implies *real* **where**

ub = ($\lambda x. x / (\alpha * (\alpha + 1)) + c2 / (\alpha + 1) + c1 * (1 / (1 - \alpha) * x + c3 * x \text{ powr } \alpha)$)

have *le*: $|\text{sum-upto } (\text{divisor-sigma } \alpha) \ x - \zeta / (\alpha + 1) * x \text{ powr } (\alpha + 1)| \leq ub \ x$

if $x \geq 1$ **for** x

proof –

have *div-le*: *real* ($a \text{ div } b$) $\leq x$ **if** $a \leq x$ **for** $a \ b :: \text{nat}$

by (*rule order.trans[OF - that(1)]*) *auto*

have *sum-upto (divisor-sigma α) x* =

sum-upto (dirichlet-prod ($\lambda n. \text{real } n \text{ powr } \alpha$) ($\lambda-. 1$)) x

by (*simp add: divisor-sigma-conv-dirichlet-prod [abs-def]*)

sum-upto-def divisor-sigma-1-left [symmetric])

also have $\dots = \text{sum-upto } (\lambda n. \sum d \mid d \text{ dvd } n. \text{real } d \text{ powr } \alpha) \ x$

by (*simp add: dirichlet-prod-def*)

also have $\dots = (\sum (n, d) \in (\text{SIGMA } n: \{n. n > 0 \ \wedge \text{real } n \leq x\}. \{d. d \text{ dvd } n\}). \text{real } d \text{ powr } \alpha)$

unfolding *sum-upto-def* **by** (*subst sum.Sigma*) *auto*

also have $\dots = (\sum (d, q) \in (SIGMA\ d:\{d.\ d > 0 \wedge real\ d \leq x\}.\ \{q.\ q > 0 \wedge real\ q \leq x / d\})).\ real\ q\ powr\ \alpha$
by (rule sum.reindex-bij-witness[of - $\lambda(d, q).\ (d * q, q)\ \lambda(n, d).\ (n\ div\ d, d)$])
(use div-le in $\langle auto\ simp:\ field-simps \rangle$)
also have $\dots = sum-upto\ (\lambda d.\ sum-upto\ (\lambda q.\ q\ powr\ \alpha)\ (x / d))\ x$
by (simp add: sum-upto-def sum.Sigma)
also have $\dots = x\ powr\ (\alpha + 1) * sum-upto\ (\lambda d.\ 1 / d\ powr\ (\alpha + 1))\ x / (\alpha + 1) +$
 $sum-upto\ (\lambda d.\ R1\ (x / d))\ x$
by (simp add: R1-def sum-upto-def sum.distrib sum-subtractf sum-divide-distrib
powr-divide sum-distrib-left)
also have $sum-upto\ (\lambda d.\ 1 / d\ powr\ (\alpha + 1))\ x = \zeta - x\ powr\ -\alpha / \alpha + R2\ x$
by (simp add: R2-def powr-minus field-simps powr-diff powr-add)
finally have eq: $sum-upto\ (divisor-sigma\ \alpha)\ x =$
 $x\ powr\ (\alpha + 1) * (\zeta - x\ powr\ -\alpha / \alpha + R2\ x) / (\alpha + 1) + sum-upto\ (\lambda d.\$
 $R1\ (x / d))\ x .$

have $sum-upto\ (divisor-sigma\ \alpha)\ x - \zeta / (\alpha + 1) * x\ powr\ (\alpha + 1) =$
 $-x / (\alpha * (\alpha + 1)) + x\ powr\ (\alpha + 1) / (\alpha + 1) * R2\ x + sum-upto$
 $(\lambda d.\ R1\ (x / d))\ x$
using $x\ \alpha$
by (subst eq, simp add: divide-simps
del: div-mult-self3 div-mult-self4 div-mult-self2 div-mult-self1)
(simp add: field-simps power2-eq-square powr-add powr-minus del: div-diff
div-add
del: div-mult-self3 div-mult-self4 div-mult-self2 div-mult-self1)
also have $|\dots| \leq ub\ x$ **unfolding** ub-def
proof (rule order.trans[OF abs-triangle-ineq] order.trans[OF abs-triangle-ineq4]
add-mono)+
have $|x\ powr\ (\alpha + 1) / (\alpha + 1) * R2\ x| = x\ powr\ (\alpha + 1) / (\alpha + 1) * |R2$
 $x|$
using $x\ \alpha$ **by** (simp add: abs-mult)
also have $\dots \leq x\ powr\ (\alpha + 1) / (\alpha + 1) * (c2 * x\ powr\ (-\alpha - 1))$
using $x\ \alpha$ **by** (intro mult-left-mono c2) auto
also have $\dots = c2 / (\alpha + 1)$
using $\alpha\ x$ **by** (simp add: field-simps powr-diff powr-minus powr-add)
finally show $|x\ powr\ (\alpha + 1) / (\alpha + 1) * R2\ x| \leq c2 / (\alpha + 1) .$
next
have $|sum-upto\ (\lambda d.\ R1\ (x / real\ d))\ x| \leq sum-upto\ (\lambda d.\ |R1\ (x / real\ d)|)\ x$
unfolding sum-upto-def **by** (rule sum-abs)
also have $\dots \leq sum-upto\ (\lambda d.\ c1 * (x / real\ d)\ powr\ \alpha)\ x$
unfolding sum-upto-def **by** (intro sum-mono c1) auto
also have $\dots = c1 * x\ powr\ \alpha * sum-upto\ (\lambda d.\ 1 / real\ d\ powr\ \alpha)\ x$
by (simp add: sum-upto-def sum-distrib-left powr-divide)
also have $sum-upto\ (\lambda d.\ 1 / real\ d\ powr\ \alpha)\ x = x\ powr\ (1 - \alpha) / (1 - \alpha) +$
 $R3\ x$
using x **by** (simp add: R3-def powr-minus field-simps)
also have $c1 * x\ powr\ \alpha * (x\ powr\ (1 - \alpha) / (1 - \alpha) + R3\ x) =$
 $c1 / (1 - \alpha) * x + c1 * x\ powr\ \alpha * R3\ x$

using x **by** (*simp add: powr-diff divide-simps*
del: div-mult-self3 div-mult-self4 div-mult-self2 div-mult-self1)
(simp add: field-simps)
also have $c1 * x \text{ powr } \alpha * R3 \ x \leq c1 * x \text{ powr } \alpha * c3$
using $x \ c1(1) \ c3(2)[\text{of } x]$ **by** (*intro mult-left-mono*) *auto*
finally show $|\text{sum-upto } (\lambda d. R1 \ (x / d)) \ x| \leq c1 * (1 / (1 - \alpha)) * x + c3$
 $* x \text{ powr } \alpha$
by (*simp add: field-simps*)
qed (*use* $\alpha \ x$ **in** *simp-all*)
finally show $|\text{sum-upto } (\text{divisor-sigma } \alpha) \ x - \zeta / (\alpha + 1) * x \text{ powr } (\alpha + 1)|$
 $\leq \text{ub } x$.
qed

have eventually $(\lambda x. |\text{sum-upto } (\text{divisor-sigma } \alpha) \ x - \zeta / (\alpha + 1) * x \text{ powr } (\alpha + 1)|$
 $\leq \text{ub } x)$ *at-top*
using *eventually-ge-at-top[of 1]* **by** *eventually-elim* (*use* le **in** *auto*)
hence eventually $(\lambda x. |\text{sum-upto } (\text{divisor-sigma } \alpha) \ x - \zeta / (\alpha + 1) * x \text{ powr } (\alpha + 1)|$
 $\leq |\text{ub } x|)$ *at-top*
by *eventually-elim linarith*
hence $(\lambda x. \text{sum-upto } (\text{divisor-sigma } \alpha) \ x - \zeta / (\alpha + 1) * x \text{ powr } (\alpha + 1)) \in O(\text{ub})$
by (*intro landau-o.bigI[of 1]*) *auto*
also have $\text{ub} \in O(\lambda x. x \text{ powr } \max 1 \ \alpha)$
using α **unfolding** *ub-def* **by** (*cases* $\alpha \geq 1$; *real-asymp*)
finally show *?thesis*
by (*subst set-minus-plus [symmetric]*)
(simp-all add: fun-diff-def algebra-simps ζ -def zeta-even-numeral)
qed

10.3 Case 3: $\alpha < 0$

Last, we consider the case of a negative exponent. We have for $\alpha > 0$:

$$\sum_{n \leq x} \sigma_{-\alpha}(n) = \zeta(\alpha + 1)x + O(R(x))$$

where $R(x) = \ln x$ if $\alpha = 1$ and $R(x) = x^{\max(0, 1-\alpha)}$ otherwise.

theorem *summatory-divisor-sigma-asymptotics-neg:*

fixes $\alpha :: \text{real}$
assumes $\alpha: \alpha > 0$
defines $\delta \equiv \max 0 \ (1 - \alpha)$
defines $\zeta \equiv \text{Re } (\text{zeta } (\alpha + 1))$
shows $\text{sum-upto } (\text{divisor-sigma } (-\alpha)) = o$ (*if* $\alpha = 1$ *then* $(\lambda x. \text{pi}^2 / 6 * x) + o$
 $O(\ln)$
else $(\lambda x. \zeta * x) + o \ O(\lambda x. x \text{ powr } \delta)$)

proof –

define Ra **where** $Ra = (\lambda x. -\text{sum-upto } (\lambda d. d \text{ powr } (-\alpha) * \text{frac } (x / d)) \ x)$
define $R1$ **where** $R1 = (\lambda x. \text{sum-upto } (\lambda d. \text{real } d \text{ powr } (-\alpha)) \ x - (x \text{ powr } (1 - \alpha) / (1 - \alpha) + \zeta))$

define $R2$ **where** $R2 = (\lambda x. \text{sum-upto } (\lambda d. d \text{ powr } (-\alpha - 1)) x - (\zeta - x \text{ powr } -\alpha / \alpha))$
define $R3$ **where** $R3 = (\lambda x. \text{sum-upto } (\lambda d. d \text{ powr } -\alpha) x - x \text{ powr } (1 - \alpha) / (1 - \alpha))$
obtain $c2$ **where** $c2: c2 > 0 \wedge x. x \geq 1 \implies |R2 x| \leq c2 * x \text{ powr } (-\alpha - 1)$
using $\text{zeta-partial-sum-le-pos[of } \alpha + 1]$ α **by** $(\text{auto simp: } \zeta\text{-def } R2\text{-def})$
define $ub :: \text{real} \implies \text{real}$ **where**
 $ub = (\lambda x. x \text{ powr } (1 - \alpha) / \alpha + c2 * x \text{ powr } -\alpha + |Ra x|)$

have $le: |\text{sum-upto } (\text{divisor-sigma } (-\alpha)) x - \zeta * x| \leq ub x$
if $x: x \geq 1$ **for** x
proof $-$
have $\text{div-le: real } (a \text{ div } b) \leq x$ **if** $a \leq x$ **for** $a b :: \text{nat}$
by $(\text{rule order.trans[OF - that(1)]})$ auto

have $\text{sum-upto } (\text{divisor-sigma } (-\alpha)) x =$
 $\text{sum-upto } (\text{dirichlet-prod } (\lambda n. \text{real } n \text{ powr } (-\alpha)) (\lambda-. 1)) x$
by $(\text{simp add: divisor-sigma-conv-dirichlet-prod [abs-def] sum-upto-def divisor-sigma-1-left [symmetric]})$
also have $\dots = \text{sum-upto } (\lambda n. \sum d \mid d \text{ dvd } n. \text{real } d \text{ powr } (-\alpha)) x$
by $(\text{simp add: dirichlet-prod-def})$
also have $\dots = (\sum (n, d) \in (\text{SIGMA } n:\{n. n > 0 \wedge \text{real } n \leq x\}. \{d. d \text{ dvd } n\}). \text{real } d \text{ powr } (-\alpha))$
unfolding sum-upto-def **by** (subst sum.Sigma) auto
also have $\dots = (\sum (d, q) \in (\text{SIGMA } d:\{d. d > 0 \wedge \text{real } d \leq x\}. \{q. q > 0 \wedge \text{real } q \leq x / d\}).$
 $\text{real } d \text{ powr } (-\alpha))$
by $(\text{rule sum.reindex-bij-witness[of - } \lambda(d, q). (d * q, d) \lambda(n, d). (d, n \text{ div } d)])$
 $(\text{use div-le in } \langle \text{auto simp: field-simps dest: dvd-imp-le} \rangle)$
also have $\dots = \text{sum-upto } (\lambda d. \text{sum-upto } (\lambda q. d \text{ powr } (-\alpha)) (x / d)) x$
by $(\text{simp add: sum-upto-def sum.Sigma [symmetric]})$
also have $\dots = \text{sum-upto } (\lambda d. d \text{ powr } (-\alpha) * \lfloor x / d \rfloor) x$
using x **by** $(\text{simp add: sum-upto-altdef mult-ac})$
also have $\dots = x * \text{sum-upto } (\lambda d. d \text{ powr } (-\alpha) / d) x + Ra x$
by $(\text{simp add: frac-def sum-distrib-left sum-distrib-right sum-subtractf sum-upto-def algebra-simps Ra-def})$
also have $\text{sum-upto } (\lambda d. d \text{ powr } (-\alpha) / d) x = \text{sum-upto } (\lambda d. d \text{ powr } (-\alpha - 1)) x$
by $(\text{simp add: powr-diff powr-minus powr-add field-simps})$
also have $\dots = \zeta - x \text{ powr } -\alpha / \alpha + R2 x$
by $(\text{simp add: R2-def})$
finally have $\text{sum-upto } (\text{divisor-sigma } (-\alpha)) x - \zeta * x = -(x \text{ powr } (1 - \alpha) / \alpha) + x * R2 x + Ra x$
using $x \alpha$ **by** $(\text{simp add: powr-diff powr-minus field-simps})$

also have $|\dots| \leq x \text{ powr } (1 - \alpha) / \alpha + c2 * x \text{ powr } -\alpha + |Ra x|$
proof $(\text{rule order.trans[OF abs-triangle-ineq] order.trans[OF abs-triangle-ineq4] add-mono})+$
from x **have** $|x * R2 x| \leq x * |R2 x|$

```

    by (simp add: abs-mult)
  also from x have ... ≤ x * (c2 * x powr (-α - 1))
    by (intro mult-left-mono c2) auto
  also have ... = c2 * x powr -α
    using x by (simp add: field-simps powr-minus powr-diff)
  finally show |x * R2 x| ≤ ... .
qed (use x α in auto)
finally show |sum-upto (divisor-sigma (-α)) x - ζ * x| ≤ ub x
  by (simp add: ub-def)
qed

have eventually (λx. |sum-upto (divisor-sigma (-α)) x - ζ * x| ≤ ub x) at-top
  using eventually-ge-at-top[of 1] by eventually-elim (use le in auto)
hence eventually (λx. |sum-upto (divisor-sigma (-α)) x - ζ * x| ≤ |ub x|) at-top
  by eventually-elim linarith
hence bigo: (λx. sum-upto (divisor-sigma (-α)) x - ζ * x) ∈ O(ub)
  by (intro landau-o.bigI[of 1]) auto

define ub' :: real ⇒ real where ub' = sum-upto (λn. real n powr -α)
have |Ra x| ≤ |ub' x| if x ≥ 1 for x
proof -
  have |Ra x| ≤ sum-upto (λn. |real n powr -α * frac (x / n)|) x
    unfolding Ra-def abs-minus sum-upto-def by (rule sum-abs)
  also have ... ≤ sum-upto (λn. real n powr -α * 1) x
    unfolding abs-mult sum-upto-def
  by (intro sum-mono mult-mono) (auto intro: less-imp-le[OF frac-lt-1])
  finally show ?thesis by (simp add: ub'-def)
qed
hence Ra ∈ O(ub')
  by (intro bigoI[of - 1] eventually-mono[OF eventually-ge-at-top[of 1]]) auto
also have ub' ∈ O(λx. if α = 1 then ln x else x powr δ)
proof (cases α = 1)
  case [simp]: True
  have sum-upto (λn. 1 / n) ∈ O(ln)
    by (intro asymp-equiv-imp-bigo harm-asymp-equiv)
  thus ?thesis by (simp add: ub'-def powr-minus field-simps)
next
  case False
  have sum-upto (λn. real n powr -α) ∈ O(λx. x powr δ)
    using assms False unfolding δ-def by (intro zeta-partial-sum-pos-bigtheta
bigthetaD1)
  thus ?thesis
    using zeta-partial-sum-neg-asymp-equiv[of α] α False by (simp add: ub'-def)
qed
finally have Ra-bigo: Ra ∈ ... .

show ?thesis
proof (cases α = 1)
  case [simp]: True

```



```

with Ra-bigo have Ra: (λx. |Ra x|) ∈ O(ln) by simp
note bigo
also have ub ∈ O(λx. ln x)
  unfolding ub-def by (intro sum-in-bigo Ra) real-asymp+
finally have sum-upto (divisor-sigma (-α)) =o (λx. (pi2 / 6) * x) +o O(ln)
  by (subst set-minus-plus [symmetric])
  (simp-all add: fun-diff-def algebra-simps ζ-def zeta-even-numeral)
thus ?thesis by (simp only: True refl if-True)
next
case False
with Ra-bigo have Ra: (λx. |Ra x|) ∈ O(λx. x powr δ) by simp
have *: (λx. x powr (1 - α) / α) ∈ O(λx. x powr δ)
  (λx. c2 * x powr - α) ∈ O(λx. x powr δ)
  unfolding δ-def using α False by (cases α > 1; real-asymp)+

note bigo
also have ub ∈ O(λx. x powr δ)
  unfolding ub-def using α False by (intro sum-in-bigo Ra *)
finally have sum-upto (divisor-sigma (-α)) =o (λx. ζ * x) +o O(λx. x powr
δ)
  by (subst set-minus-plus [symmetric])
  (simp-all add: fun-diff-def algebra-simps ζ-def zeta-even-numeral)
thus ?thesis by (simp only: False refl if-False)
qed
qed
end

```

11 Selberg's asymptotic formula

```

theory Selberg-Asymptotic-Formula
imports
  More-Dirichlet-Misc
  Prime-Number-Theorem.Prime-Counting-Functions
  Shapiro-Tauberian
  Euler-MacLaurin.Euler-MacLaurin-Landau
  Partial-Zeta-Bounds
begin

```

Following Apostol, we first show an inversion formula: Consider a function $f(x)$ for $x \in \mathbb{R}_{>0}$. Define $g(x) := \ln x \cdot \sum_{n \leq x} f(x/n)$. Then:

$$f(x) \ln x + \sum_{n \leq x} \Lambda(n) f(x/n) = \sum_{n \leq x} \mu(n) g(x/n)$$

```

locale selberg-inversion =
  fixes F G :: real ⇒ 'a :: {real-algebra-1, comm-ring-1}
  defines G ≡ (λx. of-real (ln x) * sum-upto (λn. F (x / n)) x)

```

begin

lemma *eq*:

assumes $x \geq 1$

shows $F x * \text{of-real } (\ln x) + \text{dirichlet-prod}' \text{ mangoldt } F x = \text{dirichlet-prod}' \text{ moebius-mu } G x$

proof –

have $F x * \text{of-real } (\ln x) =$

$\text{dirichlet-prod}' (\lambda n. \text{if } n = 1 \text{ then } 1 \text{ else } 0) (\lambda x. F x * \text{of-real } (\ln x)) x$

by (*subst dirichlet-prod'-one-left*) (*use* $\langle x \geq 1 \rangle$ **in** *auto*)

also have $\dots = \text{dirichlet-prod}' (\lambda n. \sum d \mid d \text{ dvd } n. \text{moebius-mu } d) (\lambda x. F x * \text{of-real } (\ln x)) x$

by (*intro dirichlet-prod'-cong refl, subst sum-moebius-mu-divisors'*) *auto*

finally have *eq1*: $F x * \text{of-real } (\ln x) = \dots$.

have *eq2*: $\text{dirichlet-prod}' \text{ mangoldt } F x =$

$\text{dirichlet-prod}' (\text{dirichlet-prod } \text{moebius-mu } (\lambda n. \text{of-real } (\ln (\text{real } n)))) F$

x

proof (*intro dirichlet-prod'-cong refl*)

fix $n :: \text{nat}$ **assume** $n : n > 0$

thus $\text{mangoldt } n = \text{dirichlet-prod } \text{moebius-mu } (\lambda n. \text{of-real } (\ln (\text{real } n))) :: 'a$ n

by (*intro moebius-inversion mangoldt-sum [symmetric]*) *auto*

qed

have $F x * \text{of-real } (\ln x) + \text{dirichlet-prod}' \text{ mangoldt } F x =$

$\text{sum-upto } (\lambda n. F (x / n) * (\sum d \mid d \text{ dvd } n.$

$\text{moebius-mu } d * \text{of-real } (\ln (x / n) + \ln (n \text{ div } d)))) x$

unfolding *eq1 eq2* **unfolding** *dirichlet-prod'-def sum-upto-def*

by (*simp add: algebra-simps sum.distrib dirichlet-prod-def sum-distrib-left sum-distrib-right*)

also have $\dots = \text{sum-upto } (\lambda n. F (x / n) * (\sum d \mid d \text{ dvd } n. \text{moebius-mu } d * \text{of-real } (\ln (x / d)))) x$

using $\langle x \geq 1 \rangle$ **by** (*intro sum-upto-cong refl arg-cong2[where* $f = \lambda x y. x * y]$ *sum.cong*)

$(\text{auto elim!} : \text{dvdE simp: ln-div ln-mult})$

also have $\dots = \text{sum-upto } (\lambda n. \sum d \mid d \text{ dvd } n. \text{moebius-mu } d * \text{of-real } (\ln (x / d)) * F (x / n)) x$

by (*simp add: sum-distrib-left sum-distrib-right mult-ac*)

also have $\dots = (\sum (n,d) \in (\text{SIGMA } n : \{n. n > 0 \wedge \text{real } n \leq x\}. \{d. d \text{ dvd } n\}). \text{moebius-mu } d * \text{of-real } (\ln (x / d)) * F (x / n))$

unfolding *sum-upto-def* **by** (*subst sum.Sigma*) (*auto simp: case-prod-unfold*)

also have $\dots = (\sum (d,q) \in (\text{SIGMA } d : \{d. d > 0 \wedge \text{real } d \leq x\}. \{q. q > 0 \wedge \text{real } q \leq x / d\}).$

$\text{moebius-mu } d * \text{of-real } (\ln (x / d)) * F (x / (q * d)))$

by (*rule sum.reindex-bij-witness[of - $\lambda(d,q). (d * q, d) \lambda(n,d). (d, n \text{ div } d)]$*)

$(\text{auto simp: Real.real-of-nat-div field-simps dest: dvd-imp-le})$

also have $\dots = \text{sum-upto } (\lambda d. \text{moebius-mu } d * \text{of-real } (\ln (x / d)) *$

$\text{sum-upto } (\lambda q. F (x / (q * d))) (x / d)) x$

by (*subst sum.Sigma [symmetric]*) (*auto simp: sum-upto-def sum-distrib-left*)

also have $\dots = \text{dirichlet-prod}' \text{ moebius-mu } G x$

by (*simp add: dirichlet-prod'-def G-def mult-ac*)
finally show *?thesis* .
qed
end

We can now show Selberg's formula

$$\psi(x) \ln x + \sum_{n \leq x} \Lambda(n) \psi(x/n) = 2x \ln x + O(x) .$$

theorem *selberg-asymptotic-formula:*

includes *prime-counting-notation*

shows $(\lambda x. \psi x * \ln x + \text{dirichlet-prod}' \text{ mangoldt } \psi x) = o$
 $(\lambda x. 2 * x * \ln x) + o O(\lambda x. x)$

proof –

define *C* :: *real* **where** [*simp*]: *C* = *euler-mascheroni*

define *F2* :: *real* \Rightarrow *real* **where** [*simp*]: *F2* = $(\lambda x. x - C - 1)$

define *G1* **where** *G1* = $(\lambda x. \ln x * \text{sum-upto } (\lambda n. \psi (x / n)) x)$

define *G2* **where** *G2* = $(\lambda x. \ln x * \text{sum-upto } (\lambda n. F2 (x / n)) x)$

interpret *F1*: *selberg-inversion* ψ *G1*

by *unfold-locales (simp-all add: G1-def)*

interpret *F2*: *selberg-inversion* *F2* *G2*

by *unfold-locales (simp-all add: G2-def)*

have *G1-bigo*: $(\lambda x. G1 x - (x * \ln x ^ 2 - x * \ln x)) \in O(\lambda x. \ln x ^ 2)$

proof –

have $(\lambda x. \ln x * (\text{sum-upto } (\lambda n. \psi (x / n)) x - x * \ln x + x)) \in O(\lambda x. \ln x * \ln x)$

by (*intro landau-o.big.mult-left sum-upto-psi-x-over-n-asymptotics*)

thus *?thesis* by (*simp add: power2-eq-square G1-def algebra-simps*)

qed

have *G2-bigo*: $(\lambda x. G2 x - (x * \ln x ^ 2 - x * \ln x)) \in O(\ln)$

proof –

define *R1* :: *real* \Rightarrow *real* **where** *R1* = $(\lambda x. x * \ln x * (\text{harm } (\text{nat } \lfloor x \rfloor) - (\ln x + C)))$

define *R2* :: *real* \Rightarrow *real* **where** *R2* = $(\lambda x. (C + 1) * \ln x * \text{frac } x)$

have $(\lambda x. G2 x - (x * \ln x ^ 2 - x * \ln x)) \in \Theta(\lambda x. R1 x + R2 x)$

proof (*intro bigthetaI-cong eventually-mono[OF eventually-ge-at-top[of 1]]*)

fix *x* :: *real* **assume** *x*: $x \geq 1$

have $G2 x = x * \ln x * \text{sum-upto } (\lambda n. 1 / n) x - (C + 1) * \lfloor x \rfloor * \ln x$

using *x* by (*simp add: G2-def sum-upto-altdef sum-subtractf*
sum-distrib-left sum-distrib-right algebra-simps)

also have $\text{sum-upto } (\lambda n. 1 / n) x = \text{harm } (\text{nat } \lfloor x \rfloor)$

using *x* **unfolding** *sum-upto-def harm-def*

by (*intro sum.cong*) (*auto simp: field-simps le-nat-iff le-floor-iff*)

also have $x * \ln x * \text{harm } (\text{nat } \lfloor x \rfloor) - (C + 1) * \lfloor x \rfloor * \ln x =$

$$x * \ln x ^ 2 - x * \ln x + R1 x + R2 x$$

by (*simp add: R1-def R2-def algebra-simps frac-def power2-eq-square*)
finally show $G2\ x - (x * \ln x \wedge^2 - x * \ln x) = R1\ x + R2\ x$ **by** *simp*
qed
also have $(\lambda x. R1\ x + R2\ x) \in O(\ln)$
proof (*intro sum-in-bigo*)
have $(\lambda x::real. \ln x - \ln (\text{nat } \lfloor x \rfloor)) \in O(\lambda x. \ln x - \ln (x - 1))$
proof (*intro bigoI[of - 1] eventually-mono[OF eventually-ge-at-top[of 2]]*)
fix $x :: real$ **assume** $x: x \geq 2$
thus norm $(\ln x - \ln (\text{nat } \lfloor x \rfloor)) \leq 1 * \text{norm } (\ln x - \ln (x - 1))$ **by** *auto*
qed
also have $(\lambda x::real. \ln x - \ln (x - 1)) \in O(\lambda x. 1 / x)$ **by** *real-asymp*
finally have *bigO-lin-floor*: $(\lambda x::real. \ln x - \ln (\text{nat } \lfloor x \rfloor)) \in O(\lambda x. 1 / x)$.

have $(\lambda x. \text{harm } (\text{nat } \lfloor x \rfloor) - (\ln (\text{nat } \lfloor x \rfloor) + C)) \in O(\lambda x. 1 / \text{nat } \lfloor x \rfloor)$
unfolding *C-def* **using** *harm-expansion-bigo-simple2*
by (*rule landau-o.big.compose*)
(auto intro!: filterlim-compose[OF filterlim-nat-sequentially filterlim-floor-sequentially])
also have $(\lambda x. 1 / \text{nat } \lfloor x \rfloor) \in O(\lambda x. 1 / x)$ **by** *real-asymp*
finally have $(\lambda x. \text{harm } (\text{nat } \lfloor x \rfloor) - (\ln (\text{nat } \lfloor x \rfloor) + C) - (\ln x - \ln (\text{nat } \lfloor x \rfloor)))$
 $\in O(\lambda x. 1 / x)$ **by** (*rule sum-in-bigo[OF -bigo-lin-floor]*)
hence $(\lambda x. \text{harm } (\text{nat } \lfloor x \rfloor) - (\ln x + C)) \in O(\lambda x. 1 / x)$ **by** (*simp add:*
algebra-simps)
hence $(\lambda x. x * \ln x * (\text{harm } (\text{nat } \lfloor x \rfloor) - (\ln x + C))) \in O(\lambda x. x * \ln x * (1$
 $/ x))$
by (*intro landau-o.big.mult-left*)
thus $R1 \in O(\ln)$ **by** (*simp add: landau-divide-simps R1-def*)
next
have $R2 \in O(\lambda x. 1 * \ln x * 1)$
unfolding *R2-def* **by** (*intro landau-o.big.mult landau-o.big-refl*) *real-asymp+*
thus $R2 \in O(\ln)$ **by** (*simp add: R2-def*)
qed
finally show $(\lambda x. G2\ x - (x * (\ln x)^2 - x * \ln x)) \in O(\ln)$.
qed
hence *G2-bigo'*: $(\lambda x. G2\ x - (x * (\ln x)^2 - x * \ln x)) \in O(\lambda x. \ln x \wedge^2)$
by (*rule landau-o.big.trans*) *real-asymp+*

— Now things become a bit hairy. In order to show that the ‘Big-O’ bound is actually valid for all $x \geq 1$, we need to show that $G1\ x - G2\ x$ is bounded on any compact interval starting at 1.

have $\exists c > 0. \forall x \geq 1. |G1\ x - G2\ x| \leq c * \text{sqrt } x$
proof (*rule bigoE-bounded-real-fun*)
have $(\lambda x. G1\ x - G2\ x) \in O(\lambda x. \ln x \wedge^2)$
using *sum-in-bigo(2)[OF G1-bigo G2-bigo]* **by** *simp*
also have $(\lambda x::real. \ln x \wedge^2) \in O(\text{sqrt})$ **by** *real-asymp*
finally show $(\lambda x. G1\ x - G2\ x) \in O(\text{sqrt})$.
next
fix $x :: real$ **assume** $x \geq 1$
thus $|\text{sqrt } x| \geq 1$ **by** *simp*

```

next
  fix b :: real assume b: b ≥ 1
  show bounded ((λx. G1 x - G2 x) ‘ {1..b})
  proof (rule boundedI, safe)
    fix x assume x: x ∈ {1..b}
    have |G1 x - G2 x| = |ln x * sum-upto (λn. ψ (x / n) - F2 (x / n)) x|
      by (simp add: G1-def G2-def sum-upto-def sum-distrib-left ring-distrib
sum-subtractf)
    also have ... = ln x * |sum-upto (λn. ψ (x / n) - F2 (x / n)) x|
      using x b by (simp add: abs-mult)
    also have |sum-upto (λn. ψ (x / n) - F2 (x / n)) x| ≤
      sum-upto (λn. |ψ (x / n) - F2 (x / n)|) x
      unfolding sum-upto-def by (rule sum-abs)
    also have ... ≤ sum-upto (λn. ψ x + (x + C + 1)) x
      unfolding sum-upto-def
    proof (intro sum-mono)
      fix n assume n: n ∈ {n. n > 0 ∧ real n ≤ x}
      hence le: x / n ≤ x / 1 by (intro divide-left-mono) auto
      thus |ψ (x / n) - F2 (x / n)| ≤ ψ x + (x + C + 1)
        unfolding F2-def using euler-mascheroni-pos x le ψ-nonneg ψ-mono[of x
/ n x]
      by (intro order.trans[OF abs-triangle-ineq]
order.trans[OF abs-triangle-ineq4] add-mono) auto
    qed
    also have ... = (ψ x + (x + C + 1)) * |x|
      using x by (simp add: sum-upto-altdef)
    also have ln x * ((ψ x + (x + C + 1)) * real-of-int [x]) ≤
      ln b * ((ψ b + (b + C + 1)) * real-of-int [b])
      using euler-mascheroni-pos x
      by (intro mult-mono add-mono order.refl ψ-mono add-nonneg-nonneg
mult-nonneg-nonneg
ψ-nonneg) (auto intro: floor-mono)
    finally show norm (G1 x - G2 x) ≤ ln b * ((ψ b + (b + C + 1)) * real-of-int
[b])
      using x by (simp add: mult-left-mono)
    qed
  qed auto
  then obtain A where A: A > 0 ∧ x. x ≥ 1 ⇒ |G1 x - G2 x| ≤ A * sqrt x
  by auto

```

— The rest of the proof now consists merely of combining some asymptotic estimates.

```

  have (λx. (ψ x - F2 x) * ln x + sum-upto (λn. mangoldt n * (ψ (x / n) - F2
(x / n))) x)
    ∈ Θ(λx. sum-upto (λn. moebius-mu n * (G1 (x / n) - G2 (x / n))) x)
  proof (intro bigthetaI-cong eventually-mono[OF eventually-ge-at-top[of 1]])
    fix x :: real assume x: x ≥ 1
    have (ψ x - F2 x) * ln x + sum-upto (λn. mangoldt n * (ψ (x / n) - F2 (x
/ n))) x =

```

$(\psi x * \text{of-real } (\ln x) + \text{dirichlet-prod}' \text{ mangoldt } \psi x) -$
 $(F2 x * \text{of-real } (\ln x) + \text{dirichlet-prod}' \text{ mangoldt } F2 x)$
by (*simp add: algebra-simps dirichlet-prod'-def sum-upto-def sum-subtractf sum.distrib*)
also have $\dots = \text{sum-upto } (\lambda n. \text{moebius-mu } n * (G1 (x / n) - G2 (x / n))) x$
unfolding *F1.eq[OF x] F2.eq[OF x]*
by (*simp add: dirichlet-prod'-def sum-upto-def sum-subtractf sum.distrib algebra-simps*)
finally show $(\psi x - F2 x) * \ln x + \text{sum-upto } (\lambda n. \text{mangoldt } n * (\psi (x/n) - F2 (x/n))) x = \dots$
qed
also have $(\lambda x. \text{sum-upto } (\lambda n. \text{moebius-mu } n * (G1 (x / n) - G2 (x / n))) x) \in$
 $O(\lambda x. A * \text{sqrt } x * \text{sum-upto } (\lambda x. x \text{ powr } (-1/2)) x)$
proof (*intro bigoI eventually-mono[OF eventually-ge-at-top[of 1]]*)
fix $x :: \text{real}$ **assume** $x \geq 1$
have $|\text{sum-upto } (\lambda n. \text{moebius-mu } n * (G1 (x / n) - G2 (x / n))) x| \leq$
 $\text{sum-upto } (\lambda n. |\text{moebius-mu } n * (G1 (x / n) - G2 (x / n))|) x$
unfolding *sum-upto-def* **by** (*rule sum-abs*)
also have $\dots \leq \text{sum-upto } (\lambda n. 1 * (A * \text{sqrt } (x / n))) x$
unfolding *sum-upto-def abs-mult* **by** (*intro A sum-mono mult-mono*) (*auto simp: moebius-mu-def*)
also have $\dots = A * \text{sqrt } x * \text{sum-upto } (\lambda x. x \text{ powr } (-1/2)) x$
using x **by** (*simp add: sum-upto-def powr-minus powr-half-sqrt sum-distrib-left sum-distrib-right real-sqrt-divide field-simps*)
also have $\dots \leq |A * \text{sqrt } x * \text{sum-upto } (\lambda x. x \text{ powr } (-1/2)) x|$ **by** *simp*
finally show $\text{norm } (\text{sum-upto } (\lambda n. \text{moebius-mu } n * (G1 (x / n) - G2 (x / n))) x) \leq$
 $1 * \text{norm } (A * \text{sqrt } x * \text{sum-upto } (\lambda x. x \text{ powr } (-1/2)) x)$ **by** *simp*
qed
also have $(\lambda x. A * \text{sqrt } x * \text{sum-upto } (\lambda x. x \text{ powr } (-1/2)) x) \in O(\lambda x. 1 * \text{sqrt } x * x \text{ powr } (1/2))$
using *zeta-partial-sum-le-pos-bigo[of 1 / 2]*
by (*intro landau-o.big.mult*) (*auto simp: max-def*)
also have $(\lambda x :: \text{real}. 1 * \text{sqrt } x * x \text{ powr } (1/2)) \in O(\lambda x. x)$
by *real-asymp*
finally have *bigO*: $(\lambda x. (\psi x - F2 x) * \ln x + \text{sum-upto } (\lambda n. \text{mangoldt } n * (\psi (x/n) - F2 (x/n))) x)$
 $\in O(\lambda x. x)$ (**is** $?h \in -$) .

let $?R = \lambda x. \text{sum-upto } (\lambda n. \text{mangoldt } n / n) x$
let $?lhs = \lambda x. \psi x * \ln x + \text{dirichlet-prod}' \text{ mangoldt } \psi x$

note *bigO*
also have $?h = (\lambda x. ?lhs x - (x * \ln x - (C + 1) * (\ln x + \psi x)) - x * ?R x)$
by (*rule ext*) (*simp add: algebra-simps dirichlet-prod'-def sum-distrib-right*
 $\psi\text{-def}$
 $\text{sum-upto-def sum-subtractf sum.distrib sum-distrib-left}$)
finally have $(\lambda x. ?lhs x - (x * \ln x - (C + 1) * (\ln x + \psi x)) - x * ?R x +$
 $x * (?R x - \ln x))$

```

      ∈ O(λx. x) (is ?h' ∈ -)
proof (rule sum-in-bigo)
  have (λx. x * (sum-upto (λn. mangoldt n / real n) x - ln x)) ∈ O(λx. x * 1)
    by (intro landau-o.big.mult-left ψ.asymptotics)
  thus (λx. x * (sum-upto (λn. mangoldt n / real n) x - ln x)) ∈ O(λx. x) by
simp
qed
also have ?h' = (λx. ?lhs x - (2 * x * ln x - (C + 1) * (ln x + ψ x)))
  by (simp add: fun-eq-iff algebra-simps)
finally have (λx. ?lhs x - (2*x*ln x - (C+1) * (ln x + ψ x)) - (C+1) * (ln
x + ψ x)) ∈ O(λx. x)
proof (rule sum-in-bigo)
  have (λx. ln x + ψ x) ∈ O(λx. x)
    by (intro sum-in-bigo bigthetaD1[OF ψ.bigtheta]) real-asymp+
  thus (λx. (C + 1) * (ln x + ψ x)) ∈ O(λx. x) by simp
qed
also have (λx. ?lhs x - (2*x*ln x - (C+1) * (ln x + ψ x)) - (C+1) * (ln x
+ ψ x)) =
      (λx. ?lhs x - 2 * x * ln x) by (simp add: algebra-simps)
finally show ?thesis
  by (subst set-minus-plus [symmetric]) (simp-all add: fun-diff-def algebra-simps)
qed
end

```

12 Consequences of the Prime Number Theorem

theory *PNT-Consequences*

imports

Elementary-Prime-Bounds
Prime-Number-Theorem.Mertens-Theorems
Prime-Number-Theorem.Prime-Counting-Functions
Moebius-Mu-Sum
Lcm-Nat-Upto
Primorial
Primes-Omega

begin

In this section, we will define a locale that assumes the Prime Number Theorem in order to explore some of its elementary consequences.

12.1 Statement and alternative forms of the PNT

locale *prime-number-theorem* =

assumes *prime-number-theorem* [*asympt-equiv-intros*]: $\pi \sim[at-top] (\lambda x. x / \ln x)$

begin

corollary *ψ-asymptotics* [*asympt-equiv-intros*]: $\vartheta \sim[at-top] (\lambda x. x)$

using *prime-number-theorem* **by** (*rule PNT1-imp-PNT4*)

corollary *ψ -asymptotics* [*asympt-equiv-intros*]: $\psi \sim[at-top] (\lambda x. x)$
using *ϑ -asymptotics PNT4-imp-PNT5* **by** *simp*

corollary *\ln - π -asymptotics* [*asympt-equiv-intros*]: $(\lambda x. \ln (\pi x)) \sim[at-top] \ln$
using *prime-number-theorem PNT1-imp-PNT1'* **by** *simp*

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

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

corollary *moebius-mu-smallo*: $\text{sum-upto moebius-mu} \in o(\lambda x. x)$
using *PNT-implies-sum-moebius-mu-sublinear ψ -asymptotics* **by** *simp*

lemma *\ln - ϑ -asymptotics*:
includes *prime-counting-notation*
shows $(\lambda x. \ln (\vartheta x) - \ln x) \in o(\lambda x. 1)$
proof –
have [*simp*]: $\vartheta 2 = \ln 2$
by (*simp add: eval- ϑ*)
have *ϑ -pos*: $\vartheta x > 0$ **if** $x \geq 2$ **for** x
proof –
have $0 < \ln (2 :: \text{real})$ **by** *simp*
also have $\dots \leq \vartheta x$
using *ϑ -mono[OF that]* **by** *simp*
finally show *?thesis* .
qed

have *nz*: *eventually* $(\lambda x. \vartheta x \neq 0 \vee x \neq 0)$ *at-top*
using *eventually-gt-at-top[of 0]* **by** *eventually-elim auto*
have *filterlim* $(\lambda x. \vartheta x / x)$ (*nhds 1*) *at-top*
using *asympt-equivD-strong[OF ϑ -asymptotics nz]* .
hence *filterlim* $(\lambda x. \ln (\vartheta x / x))$ (*nhds (ln 1)*) *at-top*
by (*rule tendsto-ln*) *auto*
also have *?this* \longleftrightarrow *filterlim* $(\lambda x. \ln (\vartheta x) - \ln x)$ (*nhds 0*) *at-top*
by (*intro filterlim-cong eventually-mono[OF eventually-ge-at-top[of 2]]*)
(auto simp: ln-div ϑ -pos)
finally show $(\lambda x. \ln (\vartheta x) - \ln x) \in o(\lambda x. 1)$
by (*intro smalloI-tendsto*) *auto*
qed

lemma *\ln - ϑ -asympt-equiv* [*asympt-equiv-intros*]:
includes *prime-counting-notation*
shows $(\lambda x. \ln (\vartheta x)) \sim[at-top] \ln$
proof (*rule smallo-imp-asympt-equiv*)

have $(\lambda x. \ln (\vartheta x) - \ln x) \in o(\lambda-. 1)$ **by** *(rule ln- ϑ -asymptotics)*
also have $(\lambda-. 1) \in O(\lambda x::real. \ln x)$ **by** *real-asymp*
finally show $(\lambda x. \ln (\vartheta x) - \ln x) \in o(\ln)$.
qed

lemma *ln-nth-prime-asymptotics:*

$(\lambda n. \ln (\text{nth-prime } n) - (\ln n + \ln (\ln n))) \in o(\lambda-. 1)$

proof –

have *filterlim* $(\lambda n. \ln (\text{nth-prime } n / (n * \ln n)))$ *(nhds (ln 1)) at-top*

by *(intro tendsto-ln asymp-equivD-strong[OF nth-prime-asymptotics])*

(auto intro!: eventually-mono[OF eventually-gt-at-top[of 1]])

also have *?this* \longleftrightarrow *filterlim* $(\lambda n. \ln (\text{nth-prime } n) - (\ln n + \ln (\ln n)))$ *(nhds 0) at-top*

using *prime-gt-0-nat[OF prime-nth-prime]*

by *(intro filterlim-cong refl eventually-mono[OF eventually-gt-at-top[of 1]])*

(auto simp: field-simps ln-mult ln-div)

finally show *?thesis* **by** *(intro smalloI-tendsto) auto*

qed

lemma *ln-nth-prime-asymp-equiv [asymp-equiv-intros]:*

$(\lambda n. \ln (\text{nth-prime } n)) \sim[at-top] \ln$

proof –

have $(\lambda n. \ln (\text{nth-prime } n) - (\ln n + \ln (\ln n))) \in o(\ln)$

using *ln-nth-prime-asymptotics* **by** *(rule landau-o.small.trans) real-asymp*

hence $(\lambda n. \ln (\text{nth-prime } n) - (\ln n + \ln (\ln n)) + \ln (\ln n)) \in o(\ln)$

by *(rule sum-in-smallo) real-asymp*

thus *?thesis* **by** *(intro smallo-imp-asymp-equiv) auto*

qed

The following versions use a little less notation.

corollary *prime-number-theorem'*: $((\lambda x. \pi x / (x / \ln x)) \longrightarrow 1)$ *at-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''*:

$(\lambda x. \text{card } \{p. \text{prime } p \wedge \text{real } p \leq x\}) \sim[at-top] (\lambda x. x / \ln x)$

proof –

have $\pi = (\lambda x. \text{card } \{p. \text{prime } p \wedge \text{real } p \leq x\})$

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

with *prime-number-theorem* **show** *?thesis* **by** *simp*

qed

corollary *prime-number-theorem'''*:

$(\lambda n. \text{card } \{p. \text{prime } p \wedge p \leq n\}) \sim[at-top] (\lambda n. \text{real } n / \ln (\text{real } n))$

proof –

have $(\lambda n. \text{card } \{p. \text{prime } p \wedge \text{real } p \leq \text{real } n\}) \sim[at-top] (\lambda n. \text{real } n / \ln (\text{real } n))$

using *prime-number-theorem''*

by (rule asymp-equiv-compose') (simp add: filterlim-real-sequentially)
 thus ?thesis by simp
 qed
 end

12.2 Existence of primes in intervals

For fixed ε , The interval $(x; \varepsilon x]$ contains a prime number for any sufficiently large x . This proof was taken from A. J. Hildebrand's lecture notes [2].

lemma (in *prime-number-theorem*) *prime-in-interval-exists*:

fixes $c :: \text{real}$
 assumes $c > 1$
 shows eventually $(\lambda x. \exists p. \text{prime } p \wedge \text{real } p \in \{x <.. c * x\})$ at-top
proof –
 from $\langle c > 1 \rangle$ have $(\lambda x. \pi (c * x) / \pi x) \sim[at-top] (\lambda x. ((c * x) / \ln (c * x)) / (x / \ln x))$
 by (intro asymp-equiv-intros asymp-equiv-compose'[OF prime-number-theorem])
 real-asymp+
 also have ... $\sim[at-top] (\lambda x. c)$
 using $\langle c > 1 \rangle$ by real-asymp
 finally have $(\lambda x. \pi (c * x) / \pi x) \sim[at-top] (\lambda a. c)$ by simp
 hence $((\lambda x. \pi (c * x) / \pi x) \longrightarrow c)$ at-top
 by (rule asymp-equivD-const)
 from this and $\langle c > 1 \rangle$ have eventually $(\lambda x. \pi (c * x) / \pi x > 1)$ at-top
 by (rule order-tendstoD)
 moreover have eventually $(\lambda x. \pi x > 0)$ at-top
 using π -at-top by (auto simp: filterlim-at-top-dense)
 ultimately show ?thesis using eventually-gt-at-top[of 1]
proof eventually-elim
 case (elim x)
 define P where $P = \{p. \text{prime } p \wedge \text{real } p \in \{x <.. c * x\}\}$
 from elim and $\langle c > 1 \rangle$ have $1 * x < c * x$ by (intro mult-strict-right-mono)
 auto
 hence $x < c * x$ by simp
 have $P = \{p. \text{prime } p \wedge \text{real } p \leq c * x\} - \{p. \text{prime } p \wedge \text{real } p \leq x\}$
 by (auto simp: P-def)
 also have $\text{card } \dots = \text{card } \{p. \text{prime } p \wedge \text{real } p \leq c * x\} - \text{card } \{p. \text{prime } p \wedge \text{real } p \leq x\}$
 using $\langle x < c * x \rangle$ by (subst card-Diff-subset) (auto intro: finite-primes-le)
 also have $\text{real } \dots = \pi (c * x) - \pi x$
 using π -mono[of x $c * x$] $\langle x < c * x \rangle$
 by (subst of-nat-diff) (auto simp: primes-pi-def prime-sum-upto-def)
 finally have $\text{real } (\text{card } P) = \pi (c * x) - \pi x$ by simp
 moreover have $\pi (c * x) - \pi x > 0$
 using elim by (auto simp: field-simps)
 ultimately have $\text{real } (\text{card } P) > 0$ by linarith
 hence $\text{card } P > 0$ by simp
 hence $P \neq \{\}$ by (intro notI) simp

```

    thus ?case by (auto simp: P-def)
  qed
qed

```

The set of rationals whose numerator and denominator are primes is dense in $\mathbb{R}_{>0}$.

lemma (in *prime-number-theorem*) *prime-fractions-dense*:

```

  fixes  $\alpha \ \varepsilon :: \text{real}$ 
  assumes  $\alpha > 0$  and  $\varepsilon > 0$ 
  obtains  $p \ q :: \text{nat}$  where prime  $p$  and prime  $q$  and dist (real  $p$  / real  $q$ )  $\alpha < \varepsilon$ 
proof -
  define  $\varepsilon'$  where  $\varepsilon' = \varepsilon / 2$ 
  from assms have  $\varepsilon' > 0$  by (simp add:  $\varepsilon'$ -def)
  have eventually ( $\lambda x. \exists p. \text{prime } p \wedge \text{real } p \in \{x <..(1 + \varepsilon' / \alpha) * x\}$ ) at-top
    using assms  $\langle \varepsilon' > 0 \rangle$  by (intro prime-in-interval-exists) (auto simp: field-simps)
  then obtain  $x0$  where  $x0: \bigwedge x. x \geq x0 \implies \exists p. \text{prime } p \wedge \text{real } p \in \{x <..(1 + \varepsilon' / \alpha) * x\}$ 
    by (auto simp: eventually-at-top-linorder)

```

```

  have  $\exists q. \text{prime } q \wedge q > \text{nat } \lfloor x0 / \alpha \rfloor$  by (rule bigger-prime)
  then obtain  $q$  where prime  $q$   $q > \text{nat } \lfloor x0 / \alpha \rfloor$  by blast
  hence real  $q \geq x0 / \alpha$  by linarith
  with  $\langle \alpha > 0 \rangle$  have  $\alpha * \text{real } q \geq x0$  by (simp add: field-simps)
  hence  $\exists p. \text{prime } p \wedge \text{real } p \in \{\alpha * \text{real } q <..(1 + \varepsilon' / \alpha) * (\alpha * \text{real } q)\}$ 
    by (intro  $x0$ )
  then obtain  $p$  where  $p: \text{prime } p$  real  $p > \alpha * \text{real } q$  real  $p \leq (1 + \varepsilon' / \alpha) * (\alpha * \text{real } q)$ 
    using assms by auto

```

```

  from  $p$   $\langle \text{prime } q \rangle$  have real  $p / \text{real } q \leq (1 + \varepsilon' / \alpha) * \alpha$ 
    using assms by (auto simp: field-simps dest: prime-gt-0-nat)
  also have  $\dots = \alpha + \varepsilon'$ 
    using assms by (simp add: field-simps)
  finally have real  $p / \text{real } q \leq \alpha + \varepsilon'$ .
  moreover from  $p$   $\langle \text{prime } q \rangle$  have real  $p / \text{real } q > \alpha$  real  $p / \text{real } q \leq (1 + \varepsilon' / \alpha) * \alpha$ 
    using assms by (auto simp: field-simps dest: prime-gt-0-nat)
  ultimately have dist (real  $p$  / real  $q$ )  $\alpha \leq \varepsilon'$ 
    by (simp add: dist-norm)
  also have  $\dots < \varepsilon$ 
    using  $\langle \varepsilon > 0 \rangle$  by (simp add:  $\varepsilon'$ -def)
  finally show ?thesis using  $\langle \text{prime } p \rangle$   $\langle \text{prime } q \rangle$  that[of  $p$   $q$ ] by blast
qed

```

12.3 The logarithm of the primorial

The PNT directly implies the asymptotics of the logarithm of the primorial function:

context *prime-number-theorem*

begin

lemma *ln-primorial-asymp-equiv* [*asymp-equiv-intros*]:
($\lambda x. \ln (\text{primorial } x)$) \sim [*at-top*] ($\lambda x. x$)
by (*auto simp: ln-primorial ϑ -asymptotics*)

lemma *ln-ln-primorial-asymp-equiv* [*asymp-equiv-intros*]:
($\lambda x. \ln (\ln (\text{primorial } x))$) \sim [*at-top*] ($\lambda x. \ln x$)
by (*auto simp: ln-primorial ln- ϑ -asymp-equiv*)

lemma *ln-primorial'-asymp-equiv* [*asymp-equiv-intros*]:
($\lambda k. \ln (\text{primorial}' k)$) \sim [*at-top*] ($\lambda k. k * \ln k$)
and *ln-ln-primorial'-asymp-equiv* [*asymp-equiv-intros*]:
($\lambda k. \ln (\ln (\text{primorial}' k))$) \sim [*at-top*] ($\lambda k. \ln k$)
and *ln-over-ln-ln-primorial'-asymp-equiv*:
($\lambda k. \ln (\text{primorial}' k) / \ln (\ln (\text{primorial}' k))$) \sim [*at-top*] ($\lambda k. k$)

proof –

have *lim1*: *filterlim* ($\lambda k. \text{real } (\text{nth-prime } (k - 1))$) *at-top at-top*
by (*rule filterlim-compose[OF filterlim-real-sequentially]*
filterlim-compose[OF nth-prime-at-top]) **+** *real-asymp*
have *lim2*: *filterlim* ($\lambda k::\text{nat}. k - 1$) *at-top at-top*
by *real-asymp*

have ($\lambda k. \ln (\text{primorial}' k)$) \sim [*at-top*] ($\lambda n. \ln (\text{primorial } (\text{nth-prime } (n - 1)))$)
by (*intro asymp-equiv-refl-ev eventually-mono[OF eventually-gt-at-top[of 0]]*)
(*auto simp: primorial'-conv-primorial*)
also have ... \sim [*at-top*] ($\lambda n. \text{nth-prime } (n - 1)$)
by (*intro asymp-equiv-compose'[OF - lim1] asymp-equiv-intros*)
also have ... \sim [*at-top*] ($\lambda n. \text{real } (n - 1) * \ln (\text{real } (n - 1))$)
by (*intro asymp-equiv-compose'[OF - lim2] asymp-equiv-intros*)
also have ... \sim [*at-top*] ($\lambda n. n * \ln n$) **by** *real-asymp*
finally show 1: ($\lambda k. \ln (\text{primorial}' k)$) \sim [*at-top*] ($\lambda k. k * \ln k$) .

have ($\lambda k. \ln (\ln (\text{primorial}' k))$) \sim [*at-top*] ($\lambda n. \ln (\ln (\text{primorial } (\text{nth-prime } (n - 1))))$)
by (*intro asymp-equiv-refl-ev eventually-mono[OF eventually-gt-at-top[of 0]]*)
(*auto simp: primorial'-conv-primorial*)
also have ... \sim [*at-top*] ($\lambda n. \ln (\text{nth-prime } (n - 1))$)
by (*intro asymp-equiv-compose'[OF - lim1] asymp-equiv-intros*)
also have ... \sim [*at-top*] ($\lambda n. \ln (\text{real } (n - 1))$)
by (*intro asymp-equiv-compose'[OF - lim2] asymp-equiv-intros*)
also have ... \sim [*at-top*] ($\lambda n. \ln n$) **by** *real-asymp*
finally show 2: ($\lambda k. \ln (\ln (\text{primorial}' k))$) \sim [*at-top*] ($\lambda k. \ln k$) .

have ($\lambda k. \ln (\text{primorial}' k) / \ln (\ln (\text{primorial}' k))$) \sim [*at-top*] ($\lambda k. (k * \ln k) / \ln k$)
by (*intro asymp-equiv-intros 1 2*)
also have ... \sim [*at-top*] ($\lambda k. k$) **by** *real-asymp*
finally show ($\lambda k. \ln (\text{primorial}' k) / \ln (\ln (\text{primorial}' k))$) \sim [*at-top*] ($\lambda k. k$) .

qed

end

12.4 Consequences of the asymptotics of ψ and ϑ

Next, we will show some consequences of $\psi(x) \sim x$ and $\vartheta(x) \sim x$. To this end, we first show generically that any function $g = e^{x+o(x)}$ is $o(c^n)$ if $c > e$ and $\omega(c^n)$ if $c < e$.

```
locale exp-asymp-equiv-linear =
  fixes f g :: real  $\Rightarrow$  real
  assumes f-asymp-equiv:  $f \sim[at-top] (\lambda x. x)$ 
  assumes g: eventually  $(\lambda x. g x = \exp (f x)) F$ 
begin
```

lemma

```
  fixes  $\varepsilon :: real$  assumes  $\varepsilon > 0$ 
  shows smallo:  $g \in o(\lambda x. \exp ((1 + \varepsilon) * x))$ 
    and smallomega:  $g \in \omega(\lambda x. \exp ((1 - \varepsilon) * x))$ 
```

proof –

```
  have  $(\lambda x. \exp (f x) / \exp ((1 + \varepsilon) * x)) \in \Theta(\lambda x. \exp (((f x - x) / x - \varepsilon) * x))$ 
    by (intro bigthetaI-cong eventually-mono[OF eventually-gt-at-top[of 1]])
      (simp-all add: divide-simps ring-distrib flip: exp-add exp-diff)
```

```
  also have  $((\lambda x. \exp (((f x - x) / x - \varepsilon) * x)) \longrightarrow 0) at-top$ 
```

```
  proof (intro filterlim-compose[OF exp-at-bot] filterlim-tendsto-neg-mult-at-bot)
```

```
    have smallo:  $(\lambda x. f x - x) \in o(\lambda x. x)$ 
```

```
    using f-asymp-equiv by (rule asymp-equiv-imp-diff-smallo)
```

```
    show  $((\lambda x. (f x - x) / x - \varepsilon) \longrightarrow 0 - \varepsilon) at-top$ 
```

```
    by (intro tendsto-diff smalloD-tendsto[OF smallo] tendsto-const)
```

```
  qed (use  $\langle \varepsilon > 0 \rangle$  in  $\langle auto simp: filterlim-ident \rangle$ )
```

```
  hence  $(\lambda x. \exp (((f x - x) / x - \varepsilon) * x)) \in o(\lambda x. 1)$ 
```

```
  by (intro smalloI-tendsto) auto
```

```
  finally have  $(\lambda x. \exp (f x)) \in o(\lambda x. \exp ((1 + \varepsilon) * x))$ 
```

```
  by (simp add: landau-divide-simps)
```

```
  also have ?this  $\longleftrightarrow g \in o(\lambda x. \exp ((1 + \varepsilon) * x))$ 
```

```
  using g by (intro landau-o.small.in-cong) (simp add: eq-commute)
```

```
  finally show  $g \in o(\lambda x. \exp ((1 + \varepsilon) * x))$  .
```

next

```
  have  $(\lambda x. \exp (f x) / \exp ((1 - \varepsilon) * x)) \in \Theta(\lambda x. \exp (((f x - x) / x + \varepsilon) * x))$ 
```

```
  by (intro bigthetaI-cong eventually-mono[OF eventually-gt-at-top[of 1]])
```

```
      (simp add: ring-distrib flip: exp-add exp-diff)
```

```
  also have filterlim  $(\lambda x. \exp (((f x - x) / x + \varepsilon) * x)) at-top at-top$ 
```

```
  proof (intro filterlim-compose[OF exp-at-top] filterlim-tendsto-pos-mult-at-top)
```

```
    have smallo:  $(\lambda x. f x - x) \in o(\lambda x. x)$ 
```

```
    using f-asymp-equiv by (rule asymp-equiv-imp-diff-smallo)
```

```
    show  $((\lambda x. (f x - x) / x + \varepsilon) \longrightarrow 0 + \varepsilon) at-top$ 
```

```
    by (intro tendsto-add smalloD-tendsto[OF smallo] tendsto-const)
```

```
  qed (use  $\langle \varepsilon > 0 \rangle$  in  $\langle auto simp: filterlim-ident \rangle$ )
```

```
  hence  $(\lambda x. \exp (((f x - x) / x + \varepsilon) * x)) \in \omega(\lambda x. 1)$ 
```

by (*simp add: filterlim-at-top-iff-smallomega*)
finally have $(\lambda x. \exp (f x)) \in \omega(\lambda x. \exp ((1 - \varepsilon) * x))$
 by (*simp add: landau-divide-simps*)
also have $?this \longleftrightarrow g \in \omega(\lambda x. \exp ((1 - \varepsilon) * x))$
 using *g* by (*intro landau-omega.small.in-cong*) (*simp add: eq-commute*)
finally show $g \in \omega(\lambda x. \exp ((1 - \varepsilon) * x))$.
qed

lemma *smallo'*:

fixes *c* :: *real* **assumes** $c > \exp 1$
shows $g \in o(\lambda x. c \text{ powr } x)$
proof –
have $c > 0$ by (*rule le-less-trans[OF - assms]*) *auto*
from $\langle c > 0 \rangle$ *assms* **have** $\exp 1 < \exp (\ln c)$
 by (*subst exp-ln*) *auto*
hence $\ln c > 1$ by (*subst (asm) exp-less-cancel-iff*)
hence $g \in o(\lambda x. \exp ((1 + (\ln c - 1)) * x))$
 using *assms* by (*intro smallo*) *auto*
also have $(\lambda x. \exp ((1 + (\ln c - 1)) * x)) = (\lambda x. c \text{ powr } x)$
 using $\langle c > 0 \rangle$ by (*simp add: powr-def mult-ac*)
finally show *?thesis* .
qed

lemma *smallomega'*:

fixes *c* :: *real* **assumes** $c \in \{0 < .. < \exp 1\}$
shows $g \in \omega(\lambda x. c \text{ powr } x)$
proof –
from *assms* **have** $\exp 1 > \exp (\ln c)$
 by (*subst exp-ln*) *auto*
hence $\ln c < 1$ by (*subst (asm) exp-less-cancel-iff*)
hence $g \in \omega(\lambda x. \exp ((1 - (1 - \ln c)) * x))$
 using *assms* by (*intro smallomega*) *auto*
also have $(\lambda x. \exp ((1 - (1 - \ln c)) * x)) = (\lambda x. c \text{ powr } x)$
 using *assms* by (*simp add: powr-def mult-ac*)
finally show *?thesis* .
qed

end

The primorial fulfils $x\# = e^{\vartheta(x)}$ and is therefore one example of this.

context *prime-number-theorem*
begin

sublocale *primorial: exp-asymp-equiv-linear* $\vartheta \lambda x. \text{real } (\text{primorial } x)$
 using *ϑ -asymptotics* by *unfold-locales* (*simp-all add: ln-primorial [symmetric]*)

end

The LCM of the first n natural numbers is equal to $e^{\psi(n)}$ and is therefore

another example.

context *prime-number-theorem*
begin

sublocale *Lcm-upto: exp-asymp-equiv-linear* ψ $\lambda x. \text{real } (Lcm \{1..nat [x]\})$
using *ψ -asymptotics* **by** *unfold-locales (simp-all flip: Lcm-upto-real-conv- ψ)*

end

12.5 Bounds on the prime ω function

Next, we will examine the asymptotic behaviour of the prime ω function $\omega(n)$, i. e. the number of distinct prime factors of n . These proofs are again taken from A. J. Hildebrand's lecture notes [2].

lemma *ln-gt-1:*

assumes $x > (3 :: \text{real})$

shows $\ln x > 1$

proof –

have $x > \exp 1$

using *exp-le assms* **by** *linarith*

hence $\ln x > \ln (\exp 1)$ **using** *assms* **by** *(subst ln-less-cancel-iff) auto*

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

qed

lemma *(in prime-number-theorem) primes-omega-primorial'-asymp-equiv:*

$(\lambda k. \text{primes-omega } (\text{primorial}' k)) \sim[\text{at-top}]$

$(\lambda k. \ln (\text{primorial}' k) / \ln (\ln (\text{primorial}' k)))$

using *ln-over-ln-ln-primorial'-asymp-equiv* **by** *(simp add: asymp-equiv-sym)*

The number of distinct prime factors of n has maximal order $\ln n / \ln \ln n$:

theorem *(in prime-number-theorem)*

limsup-primes-omega: limsup $(\lambda n. \text{primes-omega } n / (\ln n / \ln (\ln n))) = 1$

proof *(intro antisym)*

have $(\lambda k. \text{primes-omega } (\text{primorial}' k) / (\ln (\text{primorial}' k) / \ln (\ln (\text{primorial}' k)))) \longrightarrow 1$

using *primes-omega-primorial'-asymp-equiv*

by *(intro asymp-equivD-strong eventually-mono[OF eventually-gt-at-top[of 1]])*

auto

hence *limsup* $((\lambda n. \text{primes-omega } n / (\ln n / \ln (\ln n))) \circ \text{primorial}') = \text{ereal } 1$

by *(intro lim-imp-Limsup tendsto-ereal) simp-all*

hence $1 = \text{limsup } ((\lambda n. \text{ereal } (\text{primes-omega } n / (\ln n / \ln (\ln n)))) \circ \text{primorial}')$

by *(simp add: o-def)*

also have $\dots \leq \text{limsup } (\lambda n. \text{primes-omega } n / (\ln n / \ln (\ln n)))$

using *strict-mono-primorial'* **by** *(rule limsup-subseq-mono)*

finally show *limsup* $(\lambda n. \text{primes-omega } n / (\ln n / \ln (\ln n))) \geq 1$.

next

show *limsup* $(\lambda n. \text{primes-omega } n / (\ln n / \ln (\ln n))) \leq 1$

unfolding *Limsup-le-iff*

```

proof safe
  fix  $C' :: \text{ereal}$  assume  $C': C' > 1$ 
  from  $\text{ereal-dense2}$ [OF this] obtain  $C$  where  $C: C > 1 \text{ereal } C < C'$  by auto

  have  $(\lambda k. \text{primes-omega } (\text{primorial}' k) / (\ln (\text{primorial}' k) / \ln (\ln (\text{primorial}' k)))) \longrightarrow 1$ 
    (is filterlim  $?f - -$ ) using primes-omega-primorial'-asympt-equiv
    by (intro asympt-equivD-strong eventually-mono[OF eventually-gt-at-top[of 1]])
  auto
  from order-tendstoD( $\mathcal{D}$ )[OF this C(1)]
  have eventually  $(\lambda k. ?f k < C)$  at-top .
  then obtain  $k_0$  where  $k_0: \bigwedge k. k \geq k_0 \implies ?f k < C$  by (auto simp: eventually-at-top-linorder)

  have eventually  $(\lambda n::\text{nat}. \max 3 k_0 / (\ln n / \ln (\ln n)) < C)$  at-top
    using  $\langle C > 1 \rangle$  by real-asympt
  hence eventually  $(\lambda n. \text{primes-omega } n / (\ln n / \ln (\ln n)) \leq C)$  at-top
    using eventually-gt-at-top[of primorial' (max k0 3)]
  proof eventually-elim
    case (elim n)
    define  $k$  where  $k = \text{primes-omega } n$ 
    define  $m$  where  $m = \text{primorial}' k$ 
    have  $\text{primorial}' 3 \leq \text{primorial}' (\max k_0 3)$ 
      by (subst strict-mono-less-eq[OF strict-mono-primorial']) auto
    also have  $\dots < n$  by fact
    finally have  $n > 30$  by simp

  show  $?case$ 
  proof (cases  $k \geq \max 3 k_0$ )
    case True
    hence  $m \geq 30$ 
    using strict-mono-less-eq[OF strict-mono-primorial', of 3 k] by (simp add: m-def k-def)
    have  $\exp 1 \wedge 3 \leq (3 \wedge 3 :: \text{real})$ 
      using exp-le by (intro power-mono) auto
    also have  $\dots < m$  using  $\langle m \geq 30 \rangle$  by simp
    finally have  $\ln (\exp 1 \wedge 3) < \ln m$ 
      using  $\langle m \geq 30 \rangle$  by (subst ln-less-cancel-iff) auto
    hence  $\ln m > 3$  by (subst (asm) ln-realpow) auto

    have  $\text{primorial}' (\text{primes-omega } n) \leq n$ 
      using  $\langle n > 30 \rangle$  by (intro primorial'-primes-omega-le) auto
    hence  $m \leq n$  unfolding m-def k-def using elim
      by (auto simp: max-def)
    hence  $\text{primes-omega } n / (\ln n / \ln (\ln n)) \leq k / (\ln m / \ln (\ln m))$ 
      unfolding k-def using elim  $\langle m \geq 30 \rangle$  ln-gt-1[of n]  $\langle \ln m > 3 \rangle$ 
      by (intro frac-le[of primes-omega n] divide-ln-mono mult-pos-pos divide-pos-pos) auto
    also have  $\dots = ?f k$ 

```



```

    by (simp add: k-def m-def)
  also have ... < C
    by (intro k0) (use True in ‹auto simp: k-def›)
  finally show ?thesis by simp
next
case False
hence primes-omega n / (ln n / ln (ln n)) ≤ max 3 k0 / (ln n / ln (ln n))
  using elim ln-gt-1[of n] ‹n > 30›
  by (intro divide-right-mono divide-nonneg-pos) (auto simp: k-def)
also have ... < C
  using elim by simp
finally show ?thesis by simp
qed
qed
thus eventually (λn. ereal (primes-omega n / (ln n / ln (ln n))) < C') at-top
  by eventually-elim (rule le-less-trans[OF - C(2)], auto)
qed
qed

```

12.6 Bounds on the divisor function

In this section, we shall examine the growth of the divisor function $\sigma_0(n)$. In particular, we will show that $\sigma_0(n) < 2^{c \ln n / \ln \ln n}$ for all sufficiently large n if $c > 1$ and $\sigma_0(n) > 2^{c \ln n / \ln \ln n}$ for infinitely many n if $c < 1$.

An equivalent statement is that $\ln(\sigma_0(n))$ has maximal order $\ln 2 \cdot \ln n / \ln \ln n$. Following Apostol's somewhat didactic approach, we first show a generic bounding lemma for σ_0 that depends on some function f that we will specify later.

lemma *divisor-count-bound-gen:*

fixes $f :: \text{nat} \Rightarrow \text{real}$

assumes *eventually* $(\lambda n. f\ n \geq 2)$ *at-top*

defines $c \equiv (8 / \ln 2 :: \text{real})$

defines $g \equiv (\lambda n. (\ln n + c * f\ n * \ln (\ln n)) / (\ln (f\ n)))$

shows *eventually* $(\lambda n. \text{divisor-count } n < 2^{\text{powr } g\ n})$ *at-top*

proof –

include *prime-counting-notation*

have *eventually* $(\lambda n::\text{nat}. 1 + \log 2\ n \leq \ln n \wedge 2)$ *at-top* **by** *real-asymp*

thus *eventually* $(\lambda n. \text{divisor-count } n < 2^{\text{powr } g\ n})$ *at-top*

using *eventually-gt-at-top*[of 2] *assms*(1)

proof *eventually-elim*

fix $n :: \text{nat}$

assume $n: n > 2$ **and** $f\ n \geq 2$ **and** $1 + \log 2\ n \leq \ln n \wedge 2$

define Pr **where** [simp]: $Pr = \text{prime-factors } n$

define $Pr1$ **where** [simp]: $Pr1 = \{p \in Pr. p < f\ n\}$

define $Pr2$ **where** [simp]: $Pr2 = \{p \in Pr. p \geq f\ n\}$

have *exp* $1 < \text{real } n$

```

    using e-less-272 ⟨n > 2⟩ by linarith
  hence ln (exp 1) < ln (real n)
    using ⟨n > 2⟩ by (subst ln-less-cancel-iff) auto
  hence ln (ln n) > ln (ln (exp 1))
    by (subst ln-less-cancel-iff) auto
  hence ln (ln n) > 0 by simp

define S2 where S2 = (∑ p∈Pr2. multiplicity p n)
have f n ^ S2 = (∏ p∈Pr2. f n ^ multiplicity p n)
  by (simp add: S2-def power-sum)
also have ... ≤ (∏ p∈Pr2. real p ^ multiplicity p n)
  using ⟨f n ≥ 2⟩ by (intro prod-mono conjI power-mono) auto
also from ⟨n > 2⟩ have ... ≤ (∏ p∈Pr. real p ^ multiplicity p n)
  by (intro prod-mono2 one-le-power) (auto simp: in-prime-factors-iff dest:
prime-gt-0-nat)
  also have ... = n
    using ⟨n > 2⟩ by (subst prime-factorization-nat[of n]) auto
  finally have f n ^ S2 ≤ n .
  hence ln (f n ^ S2) ≤ ln n
    using n ⟨f n ≥ 2⟩ by (subst ln-le-cancel-iff) auto
  hence S2 ≤ ln n / ln (f n)
    using ⟨f n ≥ 2⟩ by (simp add: field-simps ln-realpow)

have le-twopow: Suc a ≤ 2 ^ a for a :: nat by (induction a) auto
have (∏ p∈Pr2. Suc (multiplicity p n)) ≤ (∏ p∈Pr2. 2 ^ multiplicity p n)
  by (intro prod-mono conjI le-twopow) auto
also have ... = 2 ^ S2
  by (simp add: S2-def power-sum)
also have ... = 2 powr real S2
  by (subst powr-realpow) auto
also have ... ≤ 2 powr (ln n / ln (f n))
  by (intro powr-mono ⟨S2 ≤ ln n / ln (f n)⟩) auto
finally have bound2: real (∏ p∈Pr2. Suc (multiplicity p n)) ≤ 2 powr (ln n /
ln (f n))
  by simp

have multiplicity-le: multiplicity p n ≤ log 2 n if p: p ∈ Pr for p
proof -
  from p have 2 ^ multiplicity p n ≤ p ^ multiplicity p n
    by (intro power-mono) (auto simp: in-prime-factors-iff dest: prime-gt-1-nat)
  also have ... = (∏ p∈{p}. p ^ multiplicity p n) by simp
  also from p have (∏ p∈{p}. p ^ multiplicity p n) ≤ (∏ p∈Pr. p ^ multiplicity
p n)
    by (intro dvd-imp-le prod-dvd-prod-subset)
    (auto simp: in-prime-factors-iff dest: prime-gt-0-nat)
  also have ... = n
    using n by (subst prime-factorization-nat[of n]) auto
  finally have 2 ^ multiplicity p n ≤ n .
  hence log 2 (2 ^ multiplicity p n) ≤ log 2 n

```

```

    using n by (subst log-le-cancel-iff) auto
  thus multiplicity p n ≤ log 2 n
    by (subst (asm) log-nat-power) auto
qed

have (∏ p ∈ Pr1. Suc (multiplicity p n)) = exp (∑ p ∈ Pr1. ln (multiplicity p n
+ 1))
  by (simp add: exp-sum)
also have (∑ p ∈ Pr1. ln (multiplicity p n + 1)) ≤ (∑ p ∈ Pr1. 2 * ln (ln n))
proof (intro sum-mono)
  fix p assume p: p ∈ Pr1
  have ln (multiplicity p n + 1) ≤ ln (1 + log 2 n)
    using p multiplicity-le[of p] by (subst ln-le-cancel-iff) auto
  also have ... ≤ ln (ln n ^ 2)
    using ⟨n > 2⟩ ⟨1 + log 2 n ≤ ln n ^ 2⟩
    by (subst ln-le-cancel-iff) (auto intro: add-pos-nonneg)
  also have ... = 2 * ln (ln n)
    using ⟨n > 2⟩ by (simp add: ln-realpow)
  finally show ln (multiplicity p n + 1) ≤ 2 * ln (ln n) .
qed
also have ... = 2 * ln (ln n) * card Pr1
  by simp
also have finite {p. prime p ∧ real p ≤ f n}
  by (rule finite-subset[of - {..nat [f n]}]) (auto simp: le-nat-iff le-floor-iff)
hence card Pr1 ≤ card {p. prime p ∧ real p ≤ f n}
  by (intro card-mono) auto
also have real ... = π (f n)
  by (simp add: primes-pi-def prime-sum-upto-def)
also have ... < 4 * (f n / ln (f n))
  using ⟨f n ≥ 2⟩ by (intro π-upper-bound'') auto
also have exp (2 * ln (ln (real n)) * (4 * (f n / ln (f n)))) =
  2 powr (c * f n * ln (ln n) / ln (f n))
  by (simp add: powr-def c-def)
finally have bound1: (∏ p ∈ Pr1. Suc (multiplicity p n)) <
  2 powr (c * f n * ln (ln (real n)) / ln (f n))
  using ⟨ln (ln n) > 0⟩ by (simp add: mult-strict-left-mono)

have divisor-count n = (∏ p ∈ Pr. Suc (multiplicity p n))
  using n by (subst divisor-count.prod-prime-factors') auto
also have Pr = Pr1 ∪ Pr2 by auto
also have real (∏ p ∈ ... Suc (multiplicity p n)) =
  real ((∏ p ∈ Pr1. Suc (multiplicity p n)) * (∏ p ∈ Pr2. Suc (multiplicity
p n)))
  by (subst prod.union-disjoint) auto
also have ... < 2 powr (c * f n * ln (ln (real n)) / ln (f n)) * 2 powr (ln n /
ln (f n))
  unfolding of-nat-mult
  by (intro mult-less-le-imp-less bound1 bound2) (auto intro!: prod-nonneg)

```

prod-pos)
also have $\dots = 2 \text{ powr } g \ n$
by (*simp add: g-def add-divide-distrib powr-add*)
finally show $\text{real } (\text{divisor-count } n) < 2 \text{ powr } g \ n .$
qed
qed

Now, Apostol explains that one can choose $f(n) := \ln n / (\ln \ln n)^2$ to obtain the desired bound.

proposition *divisor-count-upper-bound:*

fixes $\varepsilon :: \text{real}$
assumes $\varepsilon > 0$
shows *eventually* $(\lambda n. \text{divisor-count } n < 2 \text{ powr } ((1 + \varepsilon) * \ln n / \ln (\ln n)))$
at-top

proof –

define $c :: \text{real}$ **where** $c = 8 / \ln 2$
define $f :: \text{nat} \Rightarrow \text{real}$ **where** $f = (\lambda n. \ln n / (\ln (\ln n)) ^ 2)$
define g **where** $g = (\lambda n. (\ln n + c * f n * \ln (\ln n)) / (\ln (f n)))$

have *eventually* $(\lambda n. \text{divisor-count } n < 2 \text{ powr } g \ n)$ *at-top*
unfolding *g-def c-def f-def* **by** (*rule divisor-count-bound-gen*) *real-asymp+*
moreover have *eventually* $(\lambda n. 2 \text{ powr } g \ n < 2 \text{ powr } ((1 + \varepsilon) * \ln n / \ln (\ln n)))$ *at-top*
using $\langle \varepsilon > 0 \rangle$ **unfolding** *g-def c-def f-def* **by** *real-asymp*
ultimately show *eventually* $(\lambda n. \text{divisor-count } n < 2 \text{ powr } ((1 + \varepsilon) * \ln n / \ln (\ln n)))$ *at-top*
by *eventually-elim (rule less-trans)*
qed

Next, we will examine the ‘worst case’. Since any prime factor of n with multiplicity k contributes a factor of $k + 1$, it is intuitively clear that $\sigma_0(n)$ is largest w. r. t. n if it is a product of small distinct primes.

We show that indeed, if $n := x\#$ (where $x\#$ denotes the primorial), we have $\sigma_0(n) = 2^{\pi(x)}$, which, by the Prime Number Theorem, indeed exceeds $c \ln n / \ln \ln n$.

theorem (*in prime-number-theorem*) *divisor-count-primorial-gt:*

assumes $\varepsilon > 0$
defines $h \equiv \text{primorial}$
shows *eventually* $(\lambda x. \text{divisor-count } (h \ x) > 2 \text{ powr } ((1 - \varepsilon) * \ln (h \ x) / \ln (\ln (h \ x))))$ *at-top*

proof –

have $(\lambda x. (1 - \varepsilon) * \ln (h \ x) / \ln (\ln (h \ x))) \sim_{[at-top]} (\lambda x. (1 - \varepsilon) * \vartheta \ x / \ln (\vartheta \ x))$
by (*simp add: h-def ln-primorial*)
also have $\dots \sim_{[at-top]} (\lambda x. (1 - \varepsilon) * x / \ln x)$
by (*intro asymp-equiv-intros ϑ -asymptotics ln- ϑ -asymp-equiv*)
finally have $*$: $(\lambda x. (1 - \varepsilon) * \ln (h \ x) / \ln (\ln (h \ x))) \sim_{[at-top]} (\lambda x. (1 - \varepsilon) * x / \ln x)$

by simp
have $(\lambda x. (1 - \varepsilon) * \ln (h x) / (\ln (\ln (h x))) - (1 - \varepsilon) * x / \ln x) \in o(\lambda x. (1 - \varepsilon) * x / \ln x)$
using *asympt-equiv-imp-diff-smallo*[OF *] **by simp**
also have $?this \longleftrightarrow (\lambda x. (1 - \varepsilon) * \ln (h x) / (\ln (\ln (h x))) - x / \ln x + \varepsilon * x / \ln x)$
 $\in o(\lambda x. (1 - \varepsilon) * x / \ln x)$
by (*intro landau-o.small.in-cong eventually-mono*[OF *eventually-gt-at-top*[of 1]])
(auto simp: field-simps)
also have $(\lambda x. (1 - \varepsilon) * x / \ln x) \in O(\lambda x. x / \ln x)$
by real-asympt
finally have $(\lambda x. (1 - \varepsilon) * \ln (h x) / (\ln (\ln (h x))) - x / \ln x + \varepsilon * x / \ln x) \in o(\lambda x. x / \ln x)$.
hence $(\lambda x. (1 - \varepsilon) * \ln (h x) / (\ln (\ln (h x))) - x / \ln x + \varepsilon * x / \ln x - (\pi x - x / \ln x)) \in o(\lambda x. x / \ln x)$
by (*intro sum-in-smallo*[OF - *asympt-equiv-imp-diff-smallo*] *prime-number-theorem*)
hence $(\lambda x. (1 - \varepsilon) * \ln (h x) / (\ln (\ln (h x))) - \pi x + \varepsilon * (x / \ln x)) \in o(\lambda x. \varepsilon * (x / \ln x))$
using $\langle \varepsilon > 0 \rangle$ **by** (*subst landau-o.small.cmult*) (*simp-all add: algebra-simps*)
hence $(\lambda x. (1 - \varepsilon) * \ln (h x) / (\ln (\ln (h x))) - \pi x) \sim[at-top] (\lambda x. -\varepsilon * (x / \ln x))$
by (*intro smallo-imp-asympt-equiv*) *auto*
hence *eventually* $(\lambda x. (1 - \varepsilon) * \ln (h x) / (\ln (\ln (h x))) - \pi x < 0 \longleftrightarrow -\varepsilon * (x / \ln x) < 0)$ *at-top*
by (*rule asympt-equiv-eventually-neg-iff*)
moreover have *eventually* $(\lambda x. -\varepsilon * (x / \ln x) < 0)$ *at-top*
using $\langle \varepsilon > 0 \rangle$ **by** *real-asympt*
ultimately have *eventually* $(\lambda x. (1 - \varepsilon) * \ln (h x) / \ln (\ln (h x)) < \pi x)$ *at-top*
by *eventually-elim simp*
thus *eventually* $(\lambda x. \text{divisor-count } (h x) > 2 \text{ powr } ((1 - \varepsilon) * \ln (h x) / \ln (\ln (h x))))$ *at-top*
proof *eventually-elim*
case (*elim x*)
hence $2 \text{ powr } ((1 - \varepsilon) * \ln (h x) / \ln (\ln (h x))) < 2 \text{ powr } \pi x$
by (*intro powr-less-mono*) *auto*
thus *?case* **by** (*simp add: divisor-count-primorial h-def*)
qed
qed

Since $h(x) \rightarrow \infty$, this gives us our infinitely many values of n that exceed the bound.

corollary (*in prime-number-theorem*) *divisor-count-lower-bound*:

assumes $\varepsilon > 0$
shows *frequently* $(\lambda n. \text{divisor-count } n > 2 \text{ powr } ((1 - \varepsilon) * \ln n / \ln (\ln n)))$
at-top
proof –
define h **where** $h = \text{primorial}$

have eventually $(\lambda n. \text{divisor-count } n > 2 \text{ powr } ((1 - \varepsilon) * \ln n / \ln (\ln n)))$
 (filtermap h at-top)
using divisor-count-primorial-gt[OF assms] **by** (simp add: eventually-filtermap h-def)
hence frequently $(\lambda n. \text{divisor-count } n > 2 \text{ powr } ((1 - \varepsilon) * \ln n / \ln (\ln n)))$
 (filtermap h at-top)
by (intro eventually-frequently) (auto simp: filtermap-bot-iff)
moreover from this **and** primorial-at-top
have filtermap h at-top \leq at-top **by** (simp add: filterlim-def h-def)
ultimately show ?thesis
by (rule frequently-mono-filter)
qed

A different formulation that is not quite as tedious to prove is this one:

lemma (in prime-number-theorem) ln-divisor-count-primorial'-asympt-equiv:
 $(\lambda k. \ln (\text{divisor-count } (\text{primorial}' k))) \sim_{\text{at-top}}$
 $(\lambda k. \ln 2 * \ln (\text{primorial}' k) / \ln (\ln (\text{primorial}' k)))$
proof –
have $(\lambda k. \ln 2 * (\ln (\text{primorial}' k) / \ln (\ln (\text{primorial}' k)))) \sim_{\text{at-top}}$ $(\lambda k. \ln 2$
 $* k)$
by (intro asympt-equiv-intros ln-over-ln-ln-primorial'-asympt-equiv)
also have $\dots \sim_{\text{at-top}}$ $(\lambda k. \ln (\text{divisor-count } (\text{primorial}' k)))$
by (simp add: ln-realpow mult-ac)
finally show ?thesis **by** (simp add: asympt-equiv-sym mult-ac)
qed

It follows that the maximal order of the divisor function is $\ln 2 \cdot \ln n / \ln \ln n$.

theorem (in prime-number-theorem) limsup-divisor-count:
 $\limsup (\lambda n. \ln (\text{divisor-count } n) * \ln (\ln n) / \ln n) = \ln 2$
proof (intro antisym)
let ?h = primorial'
have $2 \wedge k = (1 :: \text{real}) \iff k = 0$ **for** $k :: \text{nat}$
using power-eq-1-iff[of 2::real k] **by** auto
hence $(\lambda k. \ln (\text{divisor-count } (?h k) / (\ln 2 * \ln (?h k) / \ln (\ln (?h k)))) \longrightarrow$
 1
using ln-divisor-count-primorial'-asympt-equiv
by (intro asympt-equivD-strong eventually-mono[OF eventually-gt-at-top[of 1]])
 (auto simp: power-eq-1-iff)
hence $(\lambda k. \ln (\text{divisor-count } (?h k) / (\ln 2 * \ln (?h k) / \ln (\ln (?h k))) * \ln 2)$
 $\longrightarrow 1 * \ln 2$
by (rule tendsto-mult) auto
hence $(\lambda k. \ln (\text{divisor-count } (?h k) / (\ln (?h k) / \ln (\ln (?h k)))) \longrightarrow \ln 2$
by simp
hence $\limsup ((\lambda n. \ln (\text{divisor-count } n) * \ln (\ln n) / \ln n) \circ \text{primorial}') = \text{ereal}$
 $(\ln 2)$
by (intro lim-imp-Limsup tendsto-ereal) simp-all
hence $\ln 2 = \limsup ((\lambda n. \text{ereal } (\ln (\text{divisor-count } n) * \ln (\ln n) / \ln n)) \circ$
 $\text{primorial}')$
by (simp add: o-def)

also have $\dots \leq \text{limsup } (\lambda n. \text{ln } (\text{divisor-count } n) * \text{ln } (\text{ln } n) / \text{ln } n)$
using *strict-mono-primorial'* **by** (*rule limsup-subseq-mono*)
finally show $\text{limsup } (\lambda n. \text{ln } (\text{divisor-count } n) * \text{ln } (\text{ln } n) / \text{ln } n) \geq \text{ln } 2 .$
next
show $\text{limsup } (\lambda n. \text{ln } (\text{divisor-count } n) * \text{ln } (\text{ln } n) / \text{ln } n) \leq \text{ln } 2$
unfolding *Limsup-le-iff*
proof safe
fix C' **assume** $C' > \text{ereal } (\text{ln } 2)$
from *ereal-dense2*[*OF this*] **obtain** C **where** $C: C > \text{ln } 2 \text{ereal } C < C'$ **by**
auto
define ε **where** $\varepsilon = (C / \text{ln } 2) - 1$
from C **have** $\varepsilon > 0$ **by** (*simp add: ε-def*)

have *eventually* $(\lambda n::\text{nat}. \text{ln } (\text{ln } n) > 0)$ *at-top* **by** *real-asymp*
hence *eventually* $(\lambda n. \text{ln } (\text{divisor-count } n) * \text{ln } (\text{ln } n) / \text{ln } n < C)$ *at-top*
using *divisor-count-upper-bound*[*OF ‹ε > 0›*] *eventually-gt-at-top*[*of 1*]
proof *eventually-elim*
case (*elim n*)
hence $\text{ln } (\text{divisor-count } n) < \text{ln } (2 \text{ powr } ((1 + \varepsilon) * \text{ln } n / \text{ln } (\text{ln } n)))$
by (*subst ln-less-cancel-iff*) *auto*
also have $\dots = (1 + \varepsilon) * \text{ln } 2 * \text{ln } n / \text{ln } (\text{ln } n)$
by (*simp add: ln-powr*)
finally have $\text{ln } (\text{divisor-count } n) * \text{ln } (\text{ln } n) / \text{ln } n < (1 + \varepsilon) * \text{ln } 2$
using *elim* **by** (*simp add: field-simps*)
also have $\dots = C$ **by** (*simp add: ε-def*)
finally show *?case* .
qed
thus *eventually* $(\lambda n. \text{ereal } (\text{ln } (\text{divisor-count } n) * \text{ln } (\text{ln } n) / \text{ln } n) < C')$ *at-top*
by *eventually-elim* (*rule less-trans*[*OF - C(2)*], *auto*)
qed
qed

12.7 Mertens' Third Theorem

In this section, we will show that

$$\prod_{p \leq x} \left(1 - \frac{1}{p}\right) = \frac{C}{\text{ln } x} + O\left(\frac{1}{\text{ln}^2 x}\right)$$

with explicit bounds for the factor in the 'Big-O'. Here, C is the following constant:

definition *third-mertens-const* :: *real* **where**

third-mertens-const =
exp $(-\sum p::\text{nat}. \text{if prime } p \text{ then } -\text{ln } (1 - 1 / \text{real } p) - 1 / \text{real } p \text{ else } 0) -$
meissel-mertens)

This constant is actually equal to $e^{-\gamma}$ where γ is the Euler–Mascheroni constant, but showing this is quite a bit of work, which we shall not do here.

lemma *third-mertens-const-pos*: $third\text{-mertens-const} > 0$
by (*simp add*: *third-mertens-const-def*)

theorem

defines $C \equiv third\text{-mertens-const}$

shows *mertens-third-theorem-strong*:

$eventually (\lambda x. |(\prod p \mid prime\ p \wedge real\ p \leq x. 1 - 1 / p) - C / \ln x| \leq 10 * C / \ln x \wedge 2)$ *at-top*

and *mertens-third-theorem*:

$(\lambda x. (\prod p \mid prime\ p \wedge real\ p \leq x. 1 - 1 / p) - C / \ln x) \in O(\lambda x. 1 / \ln x \wedge 2)$

proof -

define *Pr* **where** $Pr = (\lambda x. \{p. prime\ p \wedge real\ p \leq x\})$

define $a :: nat \Rightarrow real$

where $a = (\lambda p. if\ prime\ p\ then\ -\ln\ (1 - 1 / real\ p) - 1 / real\ p\ else\ 0)$

define *B* **where** $B = suminf\ a$

have *C-eq*: $C = exp\ (-B - meissel\text{-mertens})$

by (*simp add*: *B-def a-def third-mertens-const-def C-def*)

have *fin*: *finite* (*Pr* *x*) **for** *x*

by (*rule finite-subset*[*of* - $\{..nat\ [x]\}$]) (*auto simp*: *Pr-def le-nat-iff le-floor-iff*)

define *S* **where** $S = (\lambda x. (\sum p \in Pr\ x. \ln\ (1 - 1 / p)))$

define *R* **where** $R = (\lambda x. S\ x + \ln\ (\ln\ x) + (B + meissel\text{-mertens}))$

have *exp-S*: $exp\ (S\ x) = (\prod p \in Pr\ x. (1 - 1 / p))$ **for** *x*

proof -

have $exp\ (S\ x) = (\prod p \in Pr\ x. exp\ (\ln\ (1 - 1 / p)))$

by (*simp add*: *S-def exp-sum fin*)

also have $\dots = (\prod p \in Pr\ x. (1 - 1 / p))$

by (*intro prod.cong*) (*auto simp*: *Pr-def dest: prime-gt-1-nat*)

finally show *?thesis* .

qed

have *a-nonneg*: $a\ n \geq 0$ **for** *n*

proof (*cases prime n*)

case *True*

hence $\ln\ (1 - 1 / n) \leq -(1 / n)$

by (*intro ln-one-minus-pos-upper-bound*) (*auto dest*: *prime-gt-1-nat*)

thus *?thesis* **by** (*auto simp*: *a-def*)

qed (*auto simp*: *a-def*)

have *summable a*

proof (*rule summable-comparison-test-bigo*)

have $a \in O(\lambda n. -\ln\ (1 - 1 / n) - 1 / n)$

by (*intro bigoI*[*of* - 1]) (*auto simp*: *a-def*)

also have $(\lambda n::nat. -\ln\ (1 - 1 / n) - 1 / n) \in O(\lambda n. 1 / n \wedge 2)$

by *real-asymp*

finally show $a \in O(\lambda n. 1 / real\ n \wedge 2)$.

next


```

show summable (λn. norm (1 / real n ^ 2))
  using inverse-power-summable[of 2] by (simp add: field-simps)
qed

have eventually (λn. a n ≤ 1 / n - 1 / Suc n) at-top
proof -
  have eventually (λn::nat. -ln (1 - 1 / n) - 1 / n ≤ 1 / n - 1 / Suc n)
at-top
  by real-asymp
  thus ?thesis using eventually-gt-at-top[of 1]
  by eventually-elim (auto simp: a-def of-nat-diff field-simps)
qed
hence eventually (λm. ∀ n ≥ m. a n ≤ 1 / n - 1 / Suc n) at-top
  by (rule eventually-all-ge-at-top)
hence eventually (λx. ∀ n ≥ nat ⌊x⌋. a n ≤ 1 / n - 1 / Suc n) at-top
  by (rule eventually-compose-filterlim)
  (intro filterlim-compose[OF filterlim-nat-sequentially] filterlim-floor-sequentially)
hence eventually (λx. B - sum-upto a x ≤ 1 / x) at-top
  using eventually-ge-at-top[of 1::real]
proof eventually-elim
  case (elim x)
  have a-le: a n ≤ 1 / real n - 1 / real (Suc n) if real n ≥ x for n
  proof -
    from that and ⟨x ≥ 1⟩ have n ≥ nat ⌊x⌋ by linarith
    with elim and that show ?thesis by auto
  qed
  define m where m = Suc (nat ⌊x⌋)
  have telescope: (λn. 1 / (n + m) - 1 / (Suc n + m)) sums (1 / real (0 + m)
- 0)
  by (intro telescope-sums') real-asymp

  have B - (∑ n < m. a n) = (∑ n. a (n + m))
    unfolding B-def sum-upto-altdef m-def using ⟨summable a⟩
    by (subst suminf-split-initial-segment[of - Suc (nat ⌊x⌋)]) auto
  also have (∑ n < m. a n) = sum-upto a x
    unfolding m-def sum-upto-altdef by (intro sum.mono-neutral-right) (auto
simp: a-def)
  also have (∑ n. a (n + m)) ≤ (∑ n. 1 / (n + m) - 1 / (Suc n + m))
  proof (intro suminf-le allI)
    show summable (λn. a (n + m))
    by (rule summable-ignore-initial-segment) fact+
  next
    show summable (λn. 1 / (n + m) - 1 / (Suc n + m))
    using telescope by (rule sums-summable)
  next
    fix n :: nat
    have x ≤ n + m unfolding m-def using ⟨x ≥ 1⟩ by linarith
    thus a (n + m) ≤ 1 / (n + m) - 1 / (Suc n + m)
    using a-le[of n + m] ⟨x ≥ 1⟩ by simp

```

qed
also have $\dots = 1 / m$
using *telescope* **by** (*simp add: sums-iff*)
also have $x \leq m \ m > 0$
unfolding *m-def* **using** $\langle x \geq 1 \rangle$ **by** *linarith+*
hence $1 / m \leq 1 / x$
using $\langle x \geq 1 \rangle$ **by** (*intro divide-left-mono*) (*auto simp: m-def*)
finally show *?case* .
qed

moreover have *eventually* $(\lambda x::\text{real}. 1 / x \leq 1 / \ln x)$ *at-top* **by** *real-asymp*
ultimately have *eventually* $(\lambda x. B - \text{sum-upto } a \ x \leq 1 / \ln x)$ *at-top*
by *eventually-elim* (*rule order.trans*)

hence *eventually* $(\lambda x. |R \ x| \leq 5 / \ln x)$ *at-top*
using *eventually-ge-at-top*[*of 2*]
proof *eventually-elim*
case (*elim x*)
have $|(B - \text{sum-upto } a \ x) - (\text{prime-sum-upto } (\lambda p. 1 / p) \ x - \ln (\ln x) - \text{meissel-mertens})| \leq$
 $1 / \ln x + 4 / \ln x$
proof (*intro order.trans*[*OF abs-triangle-ineq4 add-mono*])
show $|\text{prime-sum-upto } (\lambda p. 1 / \text{real } p) \ x - \ln (\ln x) - \text{meissel-mertens}| \leq 4$
 $/ \ln x$
by (*intro mertens-second-theorem* $\langle x \geq 2 \rangle$)
have $\text{sum-upto } a \ x \leq B$
unfolding *B-def sum-upto-def* **by** (*intro sum-le-suminf* $\langle \text{summable } a \rangle$
a-nonneg ballI) *auto*
thus $|B - \text{sum-upto } a \ x| \leq 1 / \ln x$
using *elim* **by** *linarith*
qed
also have $\text{sum-upto } a \ x = \text{prime-sum-upto } (\lambda p. -\ln (1 - 1 / p) - 1 / p) \ x$
unfolding *sum-upto-def prime-sum-upto-altdef1 a-def* **by** (*intro sum.cong*
allI) *auto*
also have $\dots = -S \ x - \text{prime-sum-upto } (\lambda p. 1 / p) \ x$
by (*simp add: a-def S-def Pr-def prime-sum-upto-def sum-subtractf sum-negf*)
finally show $|R \ x| \leq 5 / \ln x$
by (*simp add: R-def*)
qed

moreover have *eventually* $(\lambda x::\text{real}. |5 / \ln x| < 1 / 2)$ *at-top* **by** *real-asymp*
ultimately have *eventually* $(\lambda x. \text{exp } (R \ x) - 1 \in \{-5 / \ln x..10 / \ln x\})$ *at-top*
using *eventually-gt-at-top*[*of 1*]
proof *eventually-elim*
case (*elim x*)
have $\text{exp } (R \ x) \leq \text{exp } (5 / \ln x)$
using *elim* **by** *simp*
also have $\dots \leq 1 + 10 / \ln x$
using *real-exp-bound-lemma*[*of 5 / ln x*] *elim* **by** (*simp add: abs-if split*:

if-splits)

finally have $le: \exp (R x) \leq 1 + 10 / \ln x .$

have $1 + (-5 / \ln x) \leq \exp (-5 / \ln x)$

by (*rule exp-ge-add-one-self*)

also have $\exp (-5 / \ln x) \leq \exp (R x)$

using *elim by simp*

finally have $\exp (R x) \geq 1 - 5 / \ln x$ **by** *simp*

with *le show ?case by simp*

qed

thus *eventually* $(\lambda x. |(\prod_{p \in Pr} x. 1 - 1 / \text{real } p) - C / \ln x| \leq 10 * C / \ln x$
 $\wedge 2)$ *at-top*

using *eventually-gt-at-top[of exp 1] eventually-gt-at-top[of 1]*

proof *eventually-elim*

case (*elim x*)

have $|\exp (R x) - 1| \leq 10 / \ln x$

using *elim by (simp add: abs-if)*

from *elim* **have** $|\exp (S x) - C / \ln x| = C / \ln x * |\exp (R x) - 1|$

by (*simp add: R-def exp-add C-eq exp-diff exp-minus field-simps*)

also have $\dots \leq C / \ln x * (10 / \ln x)$

using *elim by (intro mult-left-mono) (auto simp: C-eq)*

finally show *?case by (simp add: exp-S power2-eq-square mult-ac)*

qed

thus $(\lambda x. (\prod_{p \in Pr} x. 1 - 1 / \text{real } p) - C / \ln x) \in O(\lambda x. 1 / \ln x \wedge 2)$

by (*intro bigO[of - 10 * C] auto*)

qed

lemma *mertens-third-theorem-asymp-equiv:*

$(\lambda x. (\prod_{p | \text{prime } p \wedge \text{real } p \leq x. 1 - 1 / \text{real } p}) \sim [\text{at-top}]$

$(\lambda x. \text{third-mertens-const} / \ln x)$

by (*rule smallo-imp-asymp-equiv[OF landau-o.big-small-trans[OF mertens-third-theorem]]*)

(*use third-mertens-const-pos in real-asymp*)

We now show an equivalent version where $\prod_{p \leq x} (1 - 1/p)$ is replaced by

$\prod_{i=1}^k (1 - 1/p_i)$:

lemma *mertens-third-convert:*

assumes $n > 0$

shows $(\prod_{k < n. 1 - 1 / \text{real } (nth\text{-prime } k)} =$

$(\prod_{p | \text{prime } p \wedge p \leq nth\text{-prime } (n - 1). 1 - 1 / p}$

proof –

have $\text{primorial}' n = \text{primorial } (nth\text{-prime } (n - 1))$

using *assms by (simp add: primorial'-conv-primorial)*

also have $\text{real } (\text{totient } \dots) =$

$\text{primorial}' n * (\prod_{p | \text{prime } p \wedge p \leq nth\text{-prime } (n - 1). 1 - 1 / p}$

using *assms by (subst totient-primorial) (auto simp: primorial'-conv-primorial)*

finally show *?thesis*

by (*simp add: totient-primorial'*)

qed

lemma (in *prime-number-theorem*) *mertens-third-theorem-asymp-equiv'*:

$(\lambda n. (\prod k < n. 1 - 1 / \text{nth-prime } k)) \sim[\text{at-top}] (\lambda x. \text{third-mertens-const} / \ln x)$

proof –

have *lim*: *filterlim* $(\lambda n. \text{real } (\text{nth-prime } (n - 1))) \text{ at-top at-top}$

by (*intro filterlim-compose*[*OF filterlim-real-sequentially*]
filterlim-compose[*OF nth-prime-at-top*]) *real-asymp*

have $(\lambda n. (\prod k < n. 1 - 1 / \text{nth-prime } k)) \sim[\text{at-top}]$

$(\lambda n. (\prod p \mid \text{prime } p \wedge \text{real } p \leq \text{real } (\text{nth-prime } (n - 1)). 1 - 1 / p))$

by (*intro asymp-equiv-refl-ev eventually-mono*[*OF eventually-gt-at-top*[*of 0*]]]
(subst mertens-third-convert, auto)

also have $\dots \sim[\text{at-top}] (\lambda n. \text{third-mertens-const} / \ln (\text{real } (\text{nth-prime } (n - 1))))$

by (*intro asymp-equiv-compose'*[*OF mertens-third-theorem-asymp-equiv lim*])

also have $\dots \sim[\text{at-top}] (\lambda n. \text{third-mertens-const} / \ln (\text{real } (n - 1)))$

by (*intro asymp-equiv-intros asymp-equiv-compose'*[*OF ln-nth-prime-asymp-equiv*])

real-asymp

also have $\dots \sim[\text{at-top}] (\lambda n. \text{third-mertens-const} / \ln (\text{real } n))$

using *third-mertens-const-pos* **by** *real-asymp*

finally show *?thesis* .

qed

12.8 Bounds on Euler's totient function

Similarly to the divisor function, we will show that $\varphi(n)$ has minimal order $Cn / \ln \ln n$.

The first part is to show the lower bound:

theorem *totient-lower-bound*:

fixes $\varepsilon :: \text{real}$

assumes $\varepsilon > 0$

defines $C \equiv \text{third-mertens-const}$

shows *eventually* $(\lambda n. \text{totient } n > (1 - \varepsilon) * C * n / \ln (\ln n)) \text{ at-top}$

proof –

include *prime-counting-notation*

define $f :: \text{nat} \Rightarrow \text{nat}$ **where** $f = (\lambda n. \text{card } \{p \in \text{prime-factors } n. p > \ln n\})$

define $lb1$ **where** $lb1 = (\lambda n :: \text{nat}. (\prod p \mid \text{prime } p \wedge \text{real } p \leq \ln n. 1 - 1 / p))$

define $lb2$ **where** $lb2 = (\lambda n :: \text{nat}. (1 - 1 / \ln n) \text{powr } (\ln n / \ln (\ln n)))$

define $lb1'$ **where** $lb1' = (\lambda n :: \text{nat}. C / \ln (\ln n) - 10 * C / \ln (\ln n) ^ 2)$

have *eventually* $(\lambda n :: \text{nat}. 1 + \log 2 n \leq \ln n ^ 2) \text{ at-top}$ **by** *real-asymp*

hence *eventually* $(\lambda n. \text{totient } n / n \geq lb1 n * lb2 n) \text{ at-top}$

using *eventually-gt-at-top*[*of 2*]

proof *eventually-elim*

fix $n :: \text{nat}$

assume $n: n > 2$ **and** $1 + \log 2 n \leq \ln n ^ 2$

define Pr **where** [*simp*]: $Pr = \text{prime-factors } n$

```

define Pr1 where [simp]: Pr1 = {p∈Pr. p ≤ ln n}
define Pr2 where [simp]: Pr2 = {p∈Pr. p > ln n}

have exp 1 < real n
  using e-less-272 ⟨n > 2⟩ by linarith
hence ln (exp 1) < ln (real n)
  using ⟨n > 2⟩ by (subst ln-less-cancel-iff) auto
hence 1 < ln n by simp
hence ln (ln n) > ln (ln (exp 1))
  by (subst ln-less-cancel-iff) auto
hence ln (ln n) > 0 by simp

have ln n ^ f n ≤ (∏ p∈Pr2. ln n)
  by (simp add: f-def)
also have ... ≤ real (∏ p∈Pr2. p) unfolding of-nat-prod
  by (intro prod-mono) (auto dest: prime-gt-1-nat simp: in-prime-factors-iff)
also {
  have (∏ p∈Pr2. p) dvd (∏ p∈Pr2. p ^ multiplicity p n)
    by (intro prod-dvd-prod dvd-power) (auto simp: prime-factors-multiplicity)
  also have ... dvd (∏ p∈Pr. p ^ multiplicity p n)
    by (intro prod-dvd-prod-subset2) auto
  also have ... = n
    using ⟨n > 2⟩ by (subst (2) prime-factorization-nat[of n]) auto
  finally have (∏ p∈Pr2. p) ≤ n
    using ⟨n > 2⟩ by (intro dvd-imp-le) auto
}
finally have ln (ln n ^ f n) ≤ ln n
  using ⟨n > 2⟩ by (subst ln-le-cancel-iff) auto
also have ln (ln n ^ f n) = f n * ln (ln n)
  using ⟨n > 2⟩ by (subst ln-realpow) auto
finally have f-le: f n ≤ ln n / ln (ln n)
  using ⟨ln (ln n) > 0⟩ by (simp add: field-simps)

have (1 - 1 / ln n) powr (ln n / ln (ln n)) ≤ (1 - 1 / ln n) powr (real (f n))
  using ⟨n > 2⟩ and ⟨ln n > 1⟩ by (intro powr-mono' f-le) auto
also have ... = (∏ p∈Pr2. 1 - 1 / ln n)
  using ⟨n > 2⟩ and ⟨ln n > 1⟩ by (subst powr-realpow) (auto simp: f-def)
also have ... ≤ (∏ p∈Pr2. 1 - 1 / p)
  using ⟨n > 2⟩ and ⟨ln n > 1⟩
  by (intro prod-mono conjI diff-mono divide-left-mono) (auto dest: prime-gt-1-nat)
finally have bound2: (∏ p∈Pr2. 1 - 1 / p) ≥ lb2 n unfolding lb2-def .

have (∏ p | prime p ∧ real p ≤ ln n. inverse (1 - 1 / p)) ≥ (∏ p∈Pr1. inverse
(1 - 1 / p))
  using ⟨n > 2⟩ by (intro prod-mono2) (auto intro: finite-primes-le dest:
prime-gt-1-nat
field-simps
simp: in-prime-factors-iff)
  hence inverse (∏ p | prime p ∧ real p ≤ ln n. 1 - 1 / p) ≥ inverse (∏ p∈Pr1.
1 - 1 / p)

```

by (subst (1 2) prod-inversef [symmetric]) auto
 hence bound1: $(\prod_{p \in Pr1}. 1 - 1 / p) \geq lb1\ n$ **unfolding** lb1-def
 by (rule inverse-le-imp-le)
 (auto intro!: prod-pos simp: in-prime-factors-iff dest: prime-gt-1-nat)

have lb1 n * lb2 n $\leq (\prod_{p \in Pr1}. 1 - 1 / p) * (\prod_{p \in Pr2}. 1 - 1 / p)$
 by (intro mult-mono bound1 bound2 prod-nonneg ballI)
 (auto simp: in-prime-factors-iff lb1-def lb2-def dest: prime-gt-1-nat)
 also have ... = $(\prod_{p \in Pr1 \cup Pr2}. 1 - 1 / p)$
 by (subst prod.union-disjoint) auto
 also have $Pr1 \cup Pr2 = Pr$ **by** auto
 also have $(\prod_{p \in Pr}. 1 - 1 / p) = \text{totient } n / n$
 using n **by** (subst totient-formula2) auto
 finally show $\text{real } (\text{totient } n) / \text{real } n \geq lb1\ n * lb2\ n$
 by (simp add: lb1-def lb2-def)

qed

moreover have eventually $(\lambda n. |lb1\ n - C / \ln (\ln n)| \leq 10 * C / \ln (\ln n) \wedge$
 2) at-top
 unfolding lb1-def C-def **using** mertens-third-theorem-strong
 by (rule eventually-compose-filterlim) real-asymp

moreover have eventually $(\lambda n. (1 - \varepsilon) * C / \ln (\ln n) < lb1'\ n * lb2\ n)$ at-top
 unfolding lb1'-def lb2-def C-def **using** third-mertens-const-pos $\langle \varepsilon > 0 \rangle$ **by**
 real-asymp

ultimately show eventually $(\lambda n. \text{totient } n > (1 - \varepsilon) * C * n / \ln (\ln n))$ at-top
 using eventually-gt-at-top[of 0]
 proof eventually-elim
 case (elim n)
 have $(1 - \varepsilon) * C / \ln (\ln n) < lb1'\ n * lb2\ n$ **by** fact
 also have $lb1'\ n \leq lb1\ n$
 unfolding lb1'-def **using** elim **by** linarith
 hence $lb1'\ n * lb2\ n \leq lb1\ n * lb2\ n$
 by (intro mult-right-mono) (auto simp: lb2-def)
 also have ... $\leq \text{totient } n / n$ **by** fact
 finally show $\text{totient } n > (1 - \varepsilon) * C * n / (\ln (\ln n))$
 using $\langle n > 0 \rangle$ **by** (simp add: field-simps)

qed

qed

Next, we examine the ‘worst case’ of $\varphi(n)$ where n is the primorial of x . In this case, we have $\varphi(n) < cn / \ln \ln n$ for any $c > C$ for all sufficiently large n .

theorem (in prime-number-theorem) totient-primorial-less:

fixes $\varepsilon :: \text{real}$

defines $C \equiv \text{third-mertens-const}$ **and** $h \equiv \text{primorial}$

assumes $\varepsilon > 0$

shows eventually $(\lambda x. \text{totient } (h\ x) < (1 + \varepsilon) * C * h\ x / \ln (\ln (h\ x)))$ at-top

proof –
have $C > 0$ **by** (*simp add: C-def third-mertens-const-pos*)

have $(\lambda x. (1 + \varepsilon) * C / \ln (\ln (h x))) \sim[at-top] (\lambda x. (1 + \varepsilon) * C / \ln (\vartheta x))$
by (*simp add: ln-primorial h-def*)

also have $\dots \sim[at-top] (\lambda x. (1 + \varepsilon) * C / \ln x)$
by (*intro asymp-equiv-intros ln- ϑ -asymp-equiv*)

finally have $(\lambda x. (1 + \varepsilon) * C / \ln (\ln (h x)) - (1 + \varepsilon) * C / \ln x) \in o(\lambda x. (1 + \varepsilon) * C / \ln x)$
by (*rule asymp-equiv-imp-diff-smallo*)

also have $(\lambda x. (1 + \varepsilon) * C / \ln x) \in O(\lambda x. 1 / \ln x)$
by *real-asymp*

also have $(\lambda x. (1 + \varepsilon) * C / \ln (\ln (h x)) - (1 + \varepsilon) * C / \ln x) =$
 $(\lambda x. (1 + \varepsilon) * C / \ln (\ln (h x)) - C / \ln x - \varepsilon * C / \ln x)$
by (*simp add: algebra-simps fun-eq-iff add-divide-distrib*)

finally have $(\lambda x. (1 + \varepsilon) * C / \ln (\ln (h x)) - C / \ln x - \varepsilon * C / \ln x) \in o(\lambda x. 1 / \ln x)$.

hence $(\lambda x. (1 + \varepsilon) * C / \ln (\ln (h x)) - C / \ln x - \varepsilon * C / \ln x -$
 $((\prod_{p \in \{p. \text{prime } p \wedge \text{real } p \leq x\}}. 1 - 1 / \text{real } p) - C / \ln x)) \in o(\lambda x. 1 / \ln x)$
unfolding *C-def*

by (*rule sum-in-smallo[OF - landau-o.big-small-trans[OF mertens-third-theorem]]*)
real-asymp+

hence $(\lambda x. ((1 + \varepsilon) * C / \ln (\ln (h x)) - (\prod_{p \in \{p. \text{prime } p \wedge \text{real } p \leq x\}}. 1 - 1 / \text{real } p)) -$
 $(\varepsilon * C / \ln x)) \in o(\lambda x. 1 / \ln x)$
by (*simp add: algebra-simps*)

also have $(\lambda x. 1 / \ln x) \in O(\lambda x. \varepsilon * C / \ln x)$

using $\langle \varepsilon > 0 \rangle$ **by** (*simp add: landau-divide-simps C-def third-mertens-const-def*)

finally have $(\lambda x. (1 + \varepsilon) * C / \ln (\ln (h x)) - (\prod_{p \mid \text{prime } p \wedge \text{real } p \leq x}. 1 - 1 / p))$
 $\sim[at-top] (\lambda x. \varepsilon * C / \ln x)$
by (*rule smallo-imp-asymp-equiv*)

hence eventually $(\lambda x. (1 + \varepsilon) * C / \ln (\ln (h x)) - (\prod_{p \mid \text{prime } p \wedge \text{real } p \leq x}. 1 - 1 / p) > 0$
 $\iff \varepsilon * C / \ln x > 0)$ *at-top*
by (*rule asymp-equiv-eventually-pos-iff*)

moreover have eventually $(\lambda x. \varepsilon * C / \ln x > 0)$ *at-top*
using $\langle \varepsilon > 0 \rangle \langle C > 0 \rangle$ **by** *real-asymp*

ultimately have eventually $(\lambda x. (1 + \varepsilon) * C / \ln (\ln (h x)) >$
 $(\prod_{p \mid \text{prime } p \wedge \text{real } p \leq x}. 1 - 1 / p))$ *at-top*
by *eventually-elim auto*

thus *?thesis*

proof *eventually-elim*

case (*elim x*)

have $h x > 0$ **by** (*auto simp: h-def primorial-def intro!: prod-pos dest: prime-gt-0-nat*)

have $h x * ((1 + \varepsilon) * C / \ln (\ln (h x))) > \text{totient } (h x)$
using *elim primorial-pos[of x]* **unfolding** *h-def totient-primorial*

by (intro mult-strict-left-mono) auto
 thus ?case by (simp add: mult-ac)
 qed
 qed

It follows that infinitely many values of n exceed $cn/\ln(\ln n)$ when c is chosen larger than C .

corollary (in *prime-number-theorem*) *totient-upper-bound*:

assumes $\varepsilon > 0$
 defines $C \equiv \text{third-mertens-const}$
 shows frequently $(\lambda n. \text{totient } n < (1 + \varepsilon) * C * n / \ln (\ln n)) \text{ at-top}$
proof –
 define h where $h = \text{primorial}$
 have eventually $(\lambda n. \text{totient } n < (1 + \varepsilon) * C * n / \ln (\ln n)) (\text{filtermap } h \text{ at-top})$
 using *totient-primorial-less*[*OF assms(1)*] by (simp add: eventually-filtermap *C-def h-def*)
 hence frequently $(\lambda n. \text{totient } n < (1 + \varepsilon) * C * n / \ln (\ln n)) (\text{filtermap } h \text{ at-top})$
 by (intro eventually-frequently) (auto simp: filtermap-bot-iff)
 moreover from this and *primorial-at-top*
 have $\text{filtermap } h \text{ at-top} \leq \text{at-top}$ by (simp add: filterlim-def *h-def*)
 ultimately show ?thesis
 by (rule frequently-mono-filter)
 qed

Again, the following alternative formulation is somewhat nicer to prove:

lemma (in *prime-number-theorem*) *totient-primorial'-asympt-equiv*:

$(\lambda k. \text{totient } (\text{primorial}' k)) \sim[\text{at-top}] (\lambda k. \text{third-mertens-const} * \text{primorial}' k / \ln k)$

proof –

let $?C = \text{third-mertens-const}$
 have $(\lambda k. \text{totient } (\text{primorial}' k)) \sim[\text{at-top}] (\lambda k. \text{primorial}' k * (\prod_{i < k} 1 - 1 / \text{nth-prime } i))$
 by (subst *totient-primorial'*) auto
 also have $\dots \sim[\text{at-top}] (\lambda k. \text{primorial}' k * (?C / \ln k))$
 by (intro *asympt-equiv-intros mertens-third-theorem-asympt-equiv'*)
 finally show ?thesis by (simp add: *algebra-simps*)
 qed

lemma (in *prime-number-theorem*) *totient-primorial'-asympt-equiv'*:

$(\lambda k. \text{totient } (\text{primorial}' k)) \sim[\text{at-top}] (\lambda k. \text{third-mertens-const} * \text{primorial}' k / \ln (\ln (\text{primorial}' k)))$

proof –

let $?C = \text{third-mertens-const}$
 have $(\lambda k. \text{totient } (\text{primorial}' k)) \sim[\text{at-top}] (\lambda k. \text{third-mertens-const} * \text{primorial}' k / \ln k)$
 by (rule *totient-primorial'-asympt-equiv*)
 also have $\dots \sim[\text{at-top}] (\lambda k. \text{third-mertens-const} * \text{primorial}' k / \ln (\ln (\text{primorial}' k)))$
 by (rule *totient-primorial'-asympt-equiv*)

by (intro asymp-equiv-intros asymp-equiv-symI[OF ln-ln-primorial'-asymp-equiv])
 finally show ?thesis .
 qed

All in all, $\varphi(n)$ has minimal order $cn / \ln \ln n$:

theorem (in prime-number-theorem)
 liminf-totient: liminf ($\lambda n. \text{totient } n * \ln (\ln n) / n = \text{third-mertens-const}$
 (is - = ereal ?c)
proof (intro antisym)
 have ($\lambda k. \text{totient } (\text{primorial}' k) / (?c * \text{primorial}' k / \ln (\ln (\text{primorial}' k)))$)
 $\longrightarrow 1$
 using totient-primorial'-asymp-equiv'
 by (intro asymp-equivD-strong eventually-mono[OF eventually-gt-at-top[of 1]])
 auto
 hence ($\lambda k. \text{totient } (\text{primorial}' k) / (?c * \text{primorial}' k / \ln (\ln (\text{primorial}' k))) *$
 ?c)
 $\longrightarrow 1 * ?c$ by (intro tendsto-mult) auto
 hence ($\lambda k. \text{totient } (\text{primorial}' k) / (\text{primorial}' k / \ln (\ln (\text{primorial}' k)))$) \longrightarrow
 ?c
 using third-mertens-const-pos by simp
 hence liminf (($\lambda n. \text{totient } n * \ln (\ln n) / n$) \circ primorial') = ereal ?c
 by (intro lim-imp-Liminf tendsto-ereal) simp-all
 hence ?c = liminf (($\lambda n. \text{ereal } (\text{totient } n * \ln (\ln n) / n)$) \circ primorial')
 by (simp add: o-def)
 also have ... \geq liminf ($\lambda n. \text{totient } n * \ln (\ln n) / n$)
 using strict-mono-primorial' by (rule liminf-subseq-mono)
 finally show liminf ($\lambda n. \text{totient } n * \ln (\ln n) / n$) \leq ?c .
next
 show liminf ($\lambda n. \text{totient } n * \ln (\ln n) / n$) \geq ?c
 unfolding le-Liminf-iff
proof safe
 fix C' assume C' < ereal ?c
 from ereal-dense2[OF this] obtain C where C: C < ?c ereal C > C' by auto
 define ε where $\varepsilon = 1 - C / ?c$
 from C have $\varepsilon > 0$ using third-mertens-const-pos by (simp add: ε -def)

 have eventually ($\lambda n::\text{nat}. \ln (\ln n) > 0$) at-top by real-asymp
 hence eventually ($\lambda n. \text{totient } n * \ln (\ln n) / n > C$) at-top
 using totient-lower-bound[OF $\langle \varepsilon > 0 \rangle$] eventually-gt-at-top[of 1]
proof eventually-elim
 case (elim n)
 hence $\text{totient } n * \ln (\ln n) / n > (1 - \varepsilon) * ?c$
 by (simp add: field-simps)
 also have $(1 - \varepsilon) * ?c = C$
 using third-mertens-const-pos by (simp add: field-simps ε -def)
 finally show ?case .
 qed
 thus eventually ($\lambda n. \text{ereal } (\text{totient } n * \ln (\ln n) / n) > C'$) at-top
 by eventually-elim (rule less-trans[OF C(2)], auto)

qed
qed

end

References

- [1] T. M. Apostol. *Introduction to Analytic Number Theory*. Undergraduate Texts in Mathematics. Springer-Verlag, 1976.
- [2] A. Hildebrand. Introduction to Analytic Number Theory (lecture notes). <https://faculty.math.illinois.edu/~hildebr/ant/>.