Bertrand's postulate

Julian Biendarra, Manuel Eberl

October 11, 2017

Abstract

Bertrand's postulate is an early result on the distribution of prime numbers: For every positive integer n, there exists a prime number that lies strictly between n and $2n$.

The proof is ported from John Harrison's formalisation in HOL Light [1]. It proceeds by first showing that the property is true for all n greater than or equal to 600 and then showing that it also holds for all n below 600 by case distinction.

Contents

theory *Bertrand*
imports
Complex-Main
HOL−Number-Theory.Number-Theory
HOL−Library.Discrete
HOL−Decision-Procs.Approximation-Bounds
Pratt-Certificate.Pratt-Certificate
begin

0.1 Auxiliary facts

lemma *ln-2-le*: *ln 2* ≤ *355 / (512* :: *real)*
proof −
have *ln 2* ≤ *real-of-float (ub-ln2 12)* **by** (*rule ub-ln2*)
also have *ub-ln2 12* = *Float 5680* (− *13*) **by** *code-simp*
finally show *?thesis* **by** *simp*

qed

lemma *ln-2-ge*: *ln 2* $\geq$ (*5677 / 8192* :: *real*)
proof −
have *ln 2* $\geq$ *real-of-float* (*lb-ln2 12*) **by** (*rule lb-ln2*)
also have *lb-ln2 12* = *Float 5677* (−*13*) **by** *code-simp*
finally show *?thesis* **by** *simp*
qed

lemma *ln-2-ge'*: *ln* (*2* :: *real*) $\geq$ *2/3* **and** *ln-2-le'*: *ln* (*2* :: *real*) $\leq$ *16/23*
using *ln-2-le ln-2-ge* **by** *simp-all*

lemma *of-nat-ge-1-iff*: (*of-nat x* :: *'a* :: *linordered-semidom*) $\geq$ *1* $\longleftrightarrow$ *x* $\geq$ *1*
using *of-nat-le-iff*[*of 1 x*] **by** (*subst* (*asm*) *of-nat-1*)

lemma *floor-conv-div-nat*:
of-int (*floor* (*real m / real n*)) = *real* (*m div n*)
by (*subst floor-divide-of-nat-eq*) *simp*

lemma *frac-conv-mod-nat*:
frac (*real m / real n*) = *real* (*m mod n*) / *real n*
by (*cases n* = *0*)
(*simp-all add*: *frac-def floor-conv-div-nat field-simps of-nat-mult*
[*symmetric*] *of-nat-add* [*symmetric*] *del*: *of-nat-mult of-nat-add*)

lemma *of-nat-prod-mset*: *prod-mset* (*image-mset of-nat A*) = *of-nat* (*prod-mset A*)
by (*induction A*) *simp-all*

lemma *prod-mset-pos*: (⋀*x* :: *'a* :: *linordered-semidom*. *x* ∈# *A* ⟹ *x* > *0*) ⟹ *prod-mset A* > *0*
by (*induction A*) *simp-all*

lemma *ln-msetprod*:
assumes ⋀*x*. *x* ∈#*I* ⟹ *x* > *0*
shows (∑ *p*::*nat*∈#*I*. *ln p*) = *ln* (∏ *p*∈#*I*. *p*)
using *assms* **by** (*induction I*) (*simp-all add*: *of-nat-prod-mset ln-mult prod-mset-pos*)

lemma *ln-fact*: *ln* (*fact n*) = (∑ *d*=*1*..*n*. *ln d*)
by (*induction n*) (*simp-all add*: *ln-mult*)

lemma *overpower-lemma*:
fixes *f g* :: *real* ⇒ *real*
assumes *f a* $\leq$ *g a*
assumes ⋀*x*. *a* $\leq$ *x* ⟹ ((λ*x*. *g x* − *f x*) *has-real-derivative* (*d x*)) (*at x*)
assumes ⋀*x*. *a* $\leq$ *x* ⟹ *d x* $\geq$ *0*
assumes *a* $\leq$ *x*
shows *f x* $\leq$ *g x*
proof (*cases a* < *x*)

case *True*
with *assms* **have** $\exists z.\ z > a \wedge z < x \wedge g\ x - f\ x - (g\ a - f\ a) = (x - a) * d\ z$
by (*intro MVT2*) *auto*
then obtain *z* **where** *z*: $z > a\ z < x\ g\ x - f\ x - (g\ a - f\ a) = (x - a) * d\ z$
by *blast*
hence $f\ x = g\ x + (f\ a - g\ a) + (a - x) * d\ z$ **by** (*simp add*: *algebra-simps*)
also from *assms* **have** $f\ a - g\ a \leq 0$ **by** (*simp add*: *algebra-simps*)
also from *assms z* **have** $(a - x) * d\ z \leq 0 * d\ z$
by (*intro mult-right-mono*) *simp-all*
finally show *?thesis* **by** *simp*
qed (*insert assms*, *auto*)

0.2 Preliminary definitions

definition *primepow* :: *nat* ⇒ *bool* **where**
primepow q ⟷ (∃ *p k*. $1 \leq k \wedge$ *prime p* ∧ *q* = *p*^*k*)

definition *primepows* :: *nat* ⇒ *nat set* **where**
primepows n = {*x*::*nat*. *primepow x* ∧ *x dvd n*}

definition *primepow-even* :: *nat* ⇒ *bool* **where**
primepow-even q ⟷ (∃ *p k*. $1 \leq k \wedge$ *prime p* ∧ *q* = *p*^(2∗*k*))

definition *primepow-odd* :: *nat* ⇒ *bool* **where**
primepow-odd q ⟷ (∃ *p k*. $1 \leq k \wedge$ *prime p* ∧ *q* = *p*^(2∗*k*+1))

abbreviation *isprimedivisor* :: *nat* ⇒ *nat* ⇒ *bool* **where**
isprimedivisor q p ≡ *prime p* ∧ *p dvd q*

definition *aprimedivisor* :: *nat* ⇒ *nat* **where**
aprimedivisor q = (*LEAST p*. *isprimedivisor q p*)

definition *pre-mangoldt* :: *nat* ⇒ *nat* **where**
pre-mangoldt d = (*if primepow d then aprimedivisor d else 1*)

definition *mangoldt* :: *nat* ⇒ *real* **where**
mangoldt d = (*if primepow d then ln* (*aprimedivisor d*) *else 0*)

definition *mangoldt-even* :: *nat* ⇒ *real* **where**
mangoldt-even d = (*if primepow-even d then ln* (*aprimedivisor d*) *else 0*)

definition *mangoldt-odd* :: *nat* ⇒ *real* **where**
mangoldt-odd d = (*if primepow-odd d then ln* (*aprimedivisor d*) *else 0*)

definition *mangoldt-1* :: *nat* ⇒ *real* **where**
mangoldt-1 d = (*if prime d then ln d else 0*)

definition *psi* :: *nat* ⇒ *real* **where**
psi n = (∑ *d*=*1*..*n*. *mangoldt d*)

definition *psi-even* :: *nat* ⇒ *real* **where**
psi-even n = (∑ *d*=*1*..*n*. *mangoldt-even d*)

definition *psi-odd* :: *nat* ⇒ *real* **where**
psi-odd n = (∑ *d*=*1*..*n*. *mangoldt-odd d*)

abbreviation (*input*) *psi-even-2* :: *nat* ⇒ *real* **where**
psi-even-2 n ≡ (∑ *d*=*2*..*n*. *mangoldt-even d*)

abbreviation (*input*) *psi-odd-2* :: *nat* ⇒ *real* **where**
psi-odd-2 n ≡ (∑ *d*=*2*..*n*. *mangoldt-odd d*)

definition *theta* :: *nat* ⇒ *real* **where**
theta n = (∑ *p*=*1*..*n*. *if prime p then ln* (*real p*) *else 0*)

0.3 Properties of prime powers

lemma
assumes *n* ≠ *0 n* ≠ *1*
shows *prime-aprimedivisor*: *prime* (*aprimedivisor n*)
and *aprimedivisor-dvd*: *aprimedivisor n dvd n*
proof −
from *assms*(*2*) **have** *A*: ¬*is-unit n* **by** *auto*
from *LeastI-ex*[*OF prime-divisor-exists*[*OF assms*(*1*) *A*]]
show *prime* (*aprimedivisor n*) *aprimedivisor n dvd n*
unfolding *aprimedivisor-def* **by** (*simp-all add*: *conj-commute*)
qed

lemma *finite-primepows* [*simp*]: *n* ≠ *0* ⟹ *finite* (*primepows n*)
by (*rule finite-subset* [*OF - finite-divisors-nat*[*of n*]]) (*auto simp*: *primepows-def*)

lemma *primepow-gt-Suc-0*: *primepow n* ⟹ *n* > *Suc 0*
using *one-less-power*[*of p*::*nat* **for** *p*] **by** (*auto simp*: *primepow-def prime-nat-iff*)

lemma *aprimedivisor-of-prime* [*simp*]: *prime p* ⟹ *aprimedivisor p* = *p*
by (*rule primes-dvd-imp-eq*) (*auto intro*!: *prime-aprimedivisor aprimedivisor-dvd prime-gt-0-nat*)

lemma *aprimedivisor-primepow-power*:
assumes *primepow n k* > *0*
shows *aprimedivisor* (*n* ^ *k*) = *aprimedivisor n*
proof −
from *assms* **obtain** *p l* **where** *l*: *prime p l* ≥ *1 n* = *p* ^ *l*
by (*auto simp*: *primepow-def*)
from *assms primepow-gt-Suc-0*[*of n*]
have ∗: *prime* (*aprimedivisor* (*n* ^ *k*)) *aprimedivisor* (*n* ^ *k*) *dvd n* ^ *k*
by (*intro prime-aprimedivisor aprimedivisor-dvd*; *simp*)+
from ∗ *l* **have** *aprimedivisor* (*n* ^ *k*) *dvd p* ^ (*l* ∗ *k*) **by** (*simp add*: *power-mult*)

with *assms* ∗ *l* **have** *aprimedivisor* (*n* ^ *k*) *dvd p*
by (*subst* (*asm*) *prime-dvd-power-iff*) *simp-all*
with *l assms* **have** *aprimedivisor* (*n* ^ *k*) = *p*
by (*intro primes-dvd-imp-eq prime-aprimedivisor l*) *auto*
moreover from *l* **have** *aprimedivisor n dvd p* ^ *l*
by (*auto intro*: *aprimedivisor-dvd simp*: *prime-gt-0-nat*)
with *assms l* **have** *aprimedivisor n dvd p*
by (*subst* (*asm*) *prime-dvd-power-iff*) (*auto intro*!: *prime-aprimedivisor simp*: *prime-gt-0-nat*)
with *l assms* **have** *aprimedivisor n* = *p*
by (*intro primes-dvd-imp-eq prime-aprimedivisor l*) *auto*
ultimately show *?thesis* **by** *simp*
qed

lemma *aprimedivisor-prime-power*:
assumes *prime p k* > *0*
shows *aprimedivisor* (*p* ^ *k*) = *p*
proof −
from *assms* **have** ∗: *prime* (*aprimedivisor* (*p* ^ *k*)) *aprimedivisor* (*p* ^ *k*) *dvd p* ^ *k*
by (*intro prime-aprimedivisor aprimedivisor-dvd*; *simp add*: *prime-nat-iff*)+
from *assms* ∗ **have** *aprimedivisor* (*p* ^ *k*) *dvd p*
by (*subst* (*asm*) *prime-dvd-power-iff*) *simp-all*
with *assms* ∗ **show** *aprimedivisor* (*p* ^ *k*) = *p* **by** (*intro primes-dvd-imp-eq*)
qed

lemma *prime-factorization-primepow*:
assumes *primepow n*
shows *prime-factorization n* =
replicate-mset (*multiplicity* (*aprimedivisor n*) *n*) (*aprimedivisor n*)
using *assms*
by (*auto simp*: *primepow-def aprimedivisor-prime-power prime-factorization-prime-power*)

lemma *primepow-decompose*:
assumes *primepow n*
shows *aprimedivisor n* ^ *multiplicity* (*aprimedivisor n*) *n* = *n*
proof −
from *assms* **have** *n* ≠ *0* **by** (*intro notI*) (*auto simp*: *primepow-def*)
hence *n* = *prod-mset* (*prime-factorization n*)
by (*subst prod-mset-prime-factorization-nat*) *simp-all*
also have *prime-factorization n* =
replicate-mset (*multiplicity* (*aprimedivisor n*) *n*) (*aprimedivisor n*)
by (*intro prime-factorization-primepow assms*)
also have *prod-mset* ... = *aprimedivisor n* ^ *multiplicity* (*aprimedivisor n*) *n*
by *simp*
finally show *?thesis* **..**
qed

lemma *aprimedivisor-vimage*:

```
  assumes prime p
  shows   aprimedivisor -` {p} ∩ primepows n = {p ^ k |k. k > 0 ∧ p ^ k dvd
n}
proof safe
  fix q assume q: q ∈ primepows n
 hence q': q ≠ 0 q ≠ 1 by (auto simp: primepow-def primepows-def prime-nat-iff)
  let ?n = multiplicity (aprimedivisor q) q
  from q q' have q = aprimedivisor q ^ ?n ∧ ?n > 0 ∧ aprimedivisor q ^ ?n dvd
n
   by (auto simp: primepow-decompose primepows-def prime-multiplicity-gt-zero-iff
         prime-aprimedivisor prime-imp-prime-elem aprimedivisor-dvd)
  thus ∃k. q = aprimedivisor q ^ k ∧ k > 0 ∧ aprimedivisor q ^ k dvd n ..
next
  fix k :: nat assume k: p ^ k dvd n k > 0
  with assms show p ^ k ∈ aprimedivisor -` {p}
    by (auto simp: aprimedivisor-prime-power)
  with assms k show p ^ k ∈ primepows n
    by (auto simp: primepows-def primepow-def aprimedivisor-prime-power intro:
Suc-leI)
qed

lemma aprimedivisor-primepows-conv-prime-factorization:
  assumes [simp]: n ≠ 0
 shows  image-mset aprimedivisor (mset-set (primepows n)) = prime-factorization
n
         (is ?lhs = ?rhs)
proof (intro multiset-eqI)
  fix p :: nat
  show count ?lhs p = count ?rhs p
  proof (cases prime p)
    case False
    have p ∉# image-mset aprimedivisor (mset-set (primepows n))
    proof
      assume p ∈# image-mset aprimedivisor (mset-set (primepows n))
      then obtain q where p = aprimedivisor q q ∈ primepows n by auto
      with False prime-aprimedivisor[of q] have q = 0 ∨ q = 1 by blast
    with ‹q ∈ primepows n› show False by (auto simp: primepows-def primepow-def)
    qed
    hence count ?lhs p = 0 by (simp only: Multiset.not-in-iff)
    with False show ?thesis by (simp add: count-prime-factorization)
  next
    case True
    hence p > 1 by (auto simp: prime-nat-iff)
     have count ?lhs p = card (aprimedivisor -` {p} ∩ primepows n) by (simp
add: count-image-mset)
    also have aprimedivisor -` {p} ∩ primepows n = {p^k |k. k > 0 ∧ p ^ k dvd
n}
      using True by (rule aprimedivisor-vimage)
    also from True have ... = (λk. p ^ k) ` {0<..multiplicity p n}
```

```
    by (subst power-dvd-iff-le-multiplicity) auto
  also from ‹p > 1› have card ... = multiplicity p n
    by (subst card-image) (auto intro!: inj-onI simp: )
  also from True have ... = count (prime-factorization n) p
    by (simp add: count-prime-factorization)
  finally show ?thesis .
 qed
qed

lemma aprimedivisor:
 assumes n ≠ 1
 shows   prime (aprimedivisor n) aprimedivisor n dvd n
proof −
 from assms have ∃ p. prime p ∧ p dvd n by (rule prime-factor-nat)
 from LeastI-ex[OF this, folded aprimedivisor-def]
  show prime (aprimedivisor n) aprimedivisor n dvd n by blast+
qed

lemma aprimedivisor-gt-1:
 assumes n ≠ 1
 shows   aprimedivisor n > 1
proof −
 from assms have prime (aprimedivisor n) by (rule aprimedivisor)
 thus aprimedivisor n > 1 by (simp add: prime-nat-iff)
qed

lemma aprimedivisor-le:
 assumes n > 1
 shows   aprimedivisor n ≤ n
proof −
 from assms have aprimedivisor n dvd n by (intro aprimedivisor) simp-all
 with assms show aprimedivisor n ≤ n
  by (intro dvd-imp-le) simp-all
qed

lemma primepow-even-imp-primepow:
 assumes primepow-even n
 shows   primepow n
proof −
 from assms guess p k unfolding primepow-even-def
  by (elim exE conjE)
 thus ?thesis unfolding primepow-def
  by (force simp: primepow-def intro: exI[of - p, OF exI[of - 2∗k]])
qed

lemma primepow-odd-imp-primepow:
 assumes primepow-odd n
 shows   primepow n
proof −
```

```
  from assms guess p k unfolding primepow-odd-def
    by (elim exE conjE)
  thus ?thesis unfolding primepow-def
    by (force simp: primepow-def intro: exI[of - p, OF exI[of - 2*k]])
qed

lemma not-primepow-0 [simp]: ¬primepow 0
  by (simp add: primepow-def)

lemma not-primepow-Suc-0 [simp]: ¬primepow (Suc 0)
  using primepow-gt-Suc-0[of Suc 0] by auto

lemma aprimedivisor-primepow:
  assumes prime p p dvd n primepow n
  shows   aprimedivisor (p * n) = p aprimedivisor n = p
proof −
  define q where q = aprimedivisor n
  from assms(3) have n-gt-1: n > Suc 0 by (rule primepow-gt-Suc-0)
  with assms have q: prime q by (auto simp: q-def intro!: prime-aprimedivisor)
 from ‹primepow n› have n: n = q ^ multiplicity q n by (simp add: primepow-decompose
q-def)
  with assms have multiplicity q n ≠ 0 by (intro notI) simp
  with ‹prime p› ‹p dvd n› have p dvd q
    by (subst (asm) n) (auto intro: prime-dvd-power-nat)
  with ‹prime p› q have p = q by (intro primes-dvd-imp-eq)
  thus aprimedivisor n = p by (simp add: q-def)

  define r where r = aprimedivisor (p * n)
  with n-gt-1 assms have r: r dvd (p * n) prime r unfolding r-def
    by (intro aprimedivisor-dvd prime-aprimedivisor; simp)+
  hence r dvd q ^ Suc (multiplicity q n)
    by (subst (asm) n) (simp-all add: ‹p = q›)
  with r have r dvd q
   by (auto intro: prime-dvd-power-nat simp: prime-dvd-mult-iff dest: prime-dvd-power)
  with r q have r = q by (intro primes-dvd-imp-eq)
  thus aprimedivisor (p * n) = p by (simp add: r-def ‹p = q›)
qed

lemma primepow-power-iff:
  primepow (p ^ n) ⟷ primepow p ∧ n > 0
proof safe
  assume primepow (p ^ n)
  from primepow-gt-Suc-0[OF this] have n: n ≠ 0 by (intro notI) simp
  thus n > 0 by simp
  from ‹primepow (p ^ n)› obtain q k where *: k ≥ 1 prime q p ^ n = q ^ k
    by (auto simp: primepow-def)
  with prime-power-exp-nat[of q n p k] n obtain i where p = q ^ i by auto
  with ‹primepow (p ^ n)› have i ≠ 0 by (intro notI) simp
  with ‹p = q ^ i› ‹prime q› show primepow p
```

by (*auto simp*: *primepow-def intro*!: *exI*[*of* - *q*, *OF exI*[*of* - *i*]])
next
assume *primepow p n* > *0*
then obtain *q k* **where** ∗: *k* ≥ *1 prime q p* = *q* ^ *k* **by** (*auto simp*: *primepow-def*)
with ‹*n* > *0*› **show** *primepow* (*p* ^ *n*)
by (*auto simp*: *primepow-def power-mult intro*!: *exI*[*of* - *q*, *OF exI*[*of* - *k* ∗ *n*]])
qed

lemma *primepow-prime* [*simp*]: *prime n* ⟹ *primepow n*
by (*auto simp*: *primepow-def intro*!: *exI*[*of* - *n*, *OF exI*[*of* - *1*]])

lemma *primepow-prime-power* [*simp*]: *prime p* ⟹ *primepow* (*p* ^ *n*) ⟷ *n* > *0*
by (*simp add*: *primepow-power-iff*)

lemma *primepow-multD*:
assumes *primepow* (*a* ∗ *b*)
shows *a* = *1* ∨ *primepow a b* = *1* ∨ *primepow b*
proof −
from *assms* **obtain** *p k* **where** *k*: *k* ≥ *1 a* ∗ *b* = *p* ^ *k prime p*
unfolding *primepow-def* **by** *auto*
then obtain *i j* **where** *a* = *p* ^ *i b* = *p* ^ *j*
using *prime-power-mult-nat*[*of p a b*] **by** *blast*
with ‹*prime p*› **show** *a* = *1* ∨ *primepow a b* = *1* ∨ *primepow b* **by** *auto*
qed

lemma *mangoldt-primepow*:
prime p ⟹ *mangoldt* (*p* ^ *k*) = (*if k* > *0 then ln p else 0*)
by (*simp add*: *mangoldt-def aprimedivisor-prime-power*)

lemma *mangoldt-primepow′* [*simp*]: *prime p* ⟹ *k* > *0* ⟹ *mangoldt* (*p* ^ *k*) = *ln p*
by (*subst mangoldt-primepow*) *auto*

lemma *mangoldt-prime* [*simp*]: *prime p* ⟹ *mangoldt p* = *ln p*
using *mangoldt-primepow*[*of p 1*] **by** *simp*

lemma *primepow-mult-aprimedivisorI*:
assumes *primepow n*
shows *primepow* (*aprimedivisor n* ∗ *n*)
by (*subst* (*2*) *primepow-decompose*[*OF assms*, *symmetric*], *subst power-Suc* [*symmetric*],
subst primepow-prime-power)
(*insert assms*, *auto intro*!: *prime-aprimedivisor dest*: *primepow-gt-Suc-0*)

lemma *primepow-odd-altdef*:
primepow-odd n ⟷
primepow n ∧ *odd* (*multiplicity* (*aprimedivisor n*) *n*) ∧ *multiplicity* (*aprimedivisor n*) *n* > *1*
proof (*intro iffI conjI*; (*elim conjE*)*?*)
assume *primepow-odd n*

then obtain *p k* **where** *n*: *k* ≥ *1 prime p n* = *p* ^ (*2* ∗ *k* + *1*)
by (*auto simp*: *primepow-odd-def*)
thus *odd* (*multiplicity* (*aprimedivisor n*) *n*) *multiplicity* (*aprimedivisor n*) *n* > *1*
by (*simp-all add*: *aprimedivisor-primepow prime-elem-multiplicity-mult-distrib*)
next
assume *A*: *primepow n* **and** *B*: *odd* (*multiplicity* (*aprimedivisor n*) *n*)
and *C*: *multiplicity* (*aprimedivisor n*) *n* > *1*
from *A* **obtain** *p k* **where** *n*: *k* ≥ *1 prime p n* = *p* ^ *k*
by (*auto simp*: *primepow-def*)
with *B C* **have** *odd k k* > *1*
by (*simp-all add*: *aprimedivisor-primepow prime-elem-multiplicity-mult-distrib*)
then obtain *j* **where** *j*: *k* = *2* ∗ *j* + *1 j* > *0* **by** (*auto elim*!: *oddE*)
with *n* **show** *primepow-odd n* **by** (*auto simp*: *primepow-odd-def intro*!: *exI*[*of* - *p*, *OF exI*[*of* - *j*]])
qed (*auto dest*: *primepow-odd-imp-primepow*)

lemma *primepow-even-altdef*:
primepow-even n ⟷ *primepow n* ∧ *even* (*multiplicity* (*aprimedivisor n*) *n*)
proof (*intro iffI conjI*; (*elim conjE*)*?*)
assume *primepow-even n*
then obtain *p k* **where** *n*: *k* ≥ *1 prime p n* = *p* ^ (*2* ∗ *k*)
by (*auto simp*: *primepow-even-def*)
thus *even* (*multiplicity* (*aprimedivisor n*) *n*)
by (*simp-all add*: *aprimedivisor-primepow prime-elem-multiplicity-mult-distrib*)
next
assume *A*: *primepow n* **and** *B*: *even* (*multiplicity* (*aprimedivisor n*) *n*)
from *A* **obtain** *p k* **where** *n*: *k* ≥ *1 prime p n* = *p* ^ *k*
by (*auto simp*: *primepow-def*)
with *B* **have** *even k*
by (*simp-all add*: *aprimedivisor-primepow prime-elem-multiplicity-mult-distrib*)
then obtain *j* **where** *j*: *k* = *2* ∗ *j* **by** (*auto elim*!: *evenE*)
from *j n* **have** *j* ≠ *0* **by** (*intro notI*) *simp-all*
with *j n* **show** *primepow-even n*
by (*auto simp*: *primepow-even-def intro*!: *exI*[*of* - *p*, *OF exI*[*of* - *j*]])
qed (*auto dest*: *primepow-even-imp-primepow*)

lemma *prime-elem-aprimedivisor*: *d* > *1* ⟹ *prime-elem* (*aprimedivisor d*)
using *prime-aprimedivisor*[*of d*] **by** *simp*

lemma *aprimedivisor-gt-0* [*simp*]: *d* > *1* ⟹ *aprimedivisor d* > *0*
using *prime-aprimedivisor*[*of d*] **by** (*simp add*: *prime-gt-0-nat*)

lemma *aprimedivisor-not-zero* [*simp*]: *d* > *1* ⟹ *aprimedivisor d* ≠ *0*
using *prime-aprimedivisor*[*of d*] **by** (*simp add*: *prime-gt-0-nat*)

lemma *aprimedivisor-gt-Suc-0* [*simp*]: *d* > *1* ⟹ *aprimedivisor d* > *Suc 0*
using *prime-aprimedivisor*[*of d*] **by** (*simp add*: *prime-gt-Suc-0-nat*)

lemma *aprimedivisor-not-Suc-0* [*simp*]: *d* > *1* ⟹ *aprimedivisor d* ≠ *Suc 0*

using *aprimedivisor-gt-Suc-0*[*of d*] **by** (*intro notI*) (*simp del*: *aprimedivisor-gt-Suc-0*)

lemma *multiplicity-aprimedivisor-gt-0* [*simp*]:
d > *1* ⟹ *multiplicity* (*aprimedivisor d*) *d* > *0*
by (*subst multiplicity-gt-zero-iff*) (*auto intro*: *aprimedivisor-dvd*)

lemma *primepow-odd-mult*:
assumes *d* > *1*
shows *primepow-odd* (*aprimedivisor d* ∗ *d*) ⟷ *primepow-even d*
using *assms*
by (*auto simp*: *primepow-odd-altdef primepow-even-altdef primepow-mult-aprimedivisorI*
aprimedivisor-primepow prime-aprimedivisor aprimedivisor-dvd
prime-elem-multiplicity-mult-distrib prime-elem-aprimedivisor
dest!: *primepow-multD*)

lemma *primepowI*:
prime p ⟹ *k* ≥ *1* ⟹ *p* ^ *k* = *n* ⟹ *primepow n* ∧ *aprimedivisor n* = *p*
unfolding *primepow-def* **by** (*auto simp*: *aprimedivisor-prime-power*)

lemma *not-primepowI*:
assumes *prime p prime q p* ≠ *q p dvd n q dvd n*
shows ¬*primepow n*
using *assms* **by** (*auto dest*: *aprimedivisor-primepow*(*2*))

lemma *pre-mangoldt-primepow*:
assumes *primepow n aprimedivisor n* = *p*
shows *pre-mangoldt n* = *p*
using *assms* **by** (*simp add*: *pre-mangoldt-def*)

lemma *pre-mangoldt-notprimepow*:
assumes ¬*primepow n*
shows *pre-mangoldt n* = *1*
using *assms* **by** (*simp add*: *pre-mangoldt-def*)

lemma *not-primepow-1*: ¬*primepow 1* **by** *simp*

lemma *sum-prime-factorization-conv-sum-primepows*:
assumes *n* ≠ *0*
shows (∑ *q*∈*primepows n*. *f* (*aprimedivisor q*)) = (∑ *p*∈#*prime-factorization n*. *f p*)
proof −
from *assms* **have** *prime-factorization n* = *image-mset aprimedivisor* (*mset-set* (*primepows n*))
by (*rule aprimedivisor-primepows-conv-prime-factorization* [*symmetric*])
also have (∑ *p*∈#. . . . *f p*) = (∑ *q*∈*primepows n*. *f* (*aprimedivisor q*))
by (*simp add*: *image-mset.compositionality sum-unfold-sum-mset o-def*)
finally show *?thesis* **..**
qed

lemma *primepow-gt-0*: *primepow n* $\Longrightarrow$ *n > 0*
using *primepow-gt-Suc-0*[*of n*] **by** *simp*

lemma *multiplicity-aprimedivisor-Suc-0-iff*:
assumes *primepow n*
shows *multiplicity* (*aprimedivisor n*) *n* = *Suc 0* $\longleftrightarrow$ *prime n*
by (*subst* (*3*) *primepow-decompose* [*OF assms*, *symmetric*])
(*insert assms primepow-gt-Suc-0*[*OF assms*],
auto simp add: *prime-power-iff intro*!: *prime-aprimedivisor*)

lemma *primepow-cases*:
primepow d $\longleftrightarrow$
(*primepow-even d* $\wedge$ $\neg$ *primepow-odd d* $\wedge$ $\neg$ *prime d*) $\vee$
($\neg$ *primepow-even d* $\wedge$ *primepow-odd d* $\wedge$ $\neg$ *prime d*) $\vee$
($\neg$ *primepow-even d* $\wedge$ $\neg$ *primepow-odd d* $\wedge$ *prime d*)
by (*auto simp*: *primepow-even-altdef primepow-odd-altdef multiplicity-aprimedivisor-Suc-0-iff*
elim!: *oddE intro*!: *Nat.gr0I*)

0.4 Deriving a recurrence for the psi function

lemma *ln-fact-bounds*:
assumes *n > 0*
shows *abs*(*ln* (*fact n*) − *n* ∗ *ln n* + *n*) ≤ *1* + *ln n*
proof −
have ∀ *n*∈{*0*<..}. ∃ *z*>*real n*. *z* < *real* (*n* + *1*) ∧ *real* (*n* + *1*) ∗ *ln* (*real* (*n* + *1*)) −
real n ∗ *ln* (*real n*) = (*real* (*n* + *1*) − *real n*) ∗ (*ln z* + *1*)
by (*intro ballI MVT2*) (*auto intro*!: *derivative-eq-intros*)
hence ∀ *n*∈{*0*<..}. ∃ *z*>*real n*. *z* < *real* (*n* + *1*) ∧ *real* (*n* + *1*) ∗ *ln* (*real* (*n* + *1*)) −
real n ∗ *ln* (*real n*) = (*ln z* + *1*) **by** (*simp add*: *algebra-simps*)
from *bchoice*[*OF this*] **obtain** *k* :: *nat* ⇒ *real*
where *lb*: *real n* < *k n* **and** *ub*: *k n* < *real* (*n* + *1*) **and**
mvt: *real* (*n*+*1*) ∗ *ln* (*real* (*n*+*1*)) − *real n* ∗ *ln* (*real n*) = *ln* (*k n*) + *1*
if *n > 0* **for** *n*::*nat* **by** *blast*
have ∗: (*n* + *1*) ∗ *ln* (*n* + *1*) = (∑ *i*=*1*..*n*. *ln*(*k i*) + *1*) **for** *n*::*nat*
proof (*induction n*)
case (*Suc n*)
have (∑ *i* = *1*..*n*+*1*. *ln* (*k i*) + *1*) = (∑ *i* = *1*..*n*. *ln* (*k i*) + *1*) + *ln* (*k* (*n*+*1*)) + *1*
by *simp*
also from *Suc.IH* **have** (∑ *i* = *1*..*n*. *ln* (*k i*) + *1*) = *real* (*n*+*1*) ∗ *ln* (*real* (*n*+*1*)) **..**
also from *mvt*[*of n*+*1*] **have** ... = *real* (*n*+*2*) ∗ *ln* (*real* (*n*+*2*)) − *ln* (*k* (*n*+*1*)) − *1*
by *simp*
finally show *?case*
by *simp*

```
  qed simp
  have **: abs((∑ i=1..n+1. ln i) − ((n+1) * ln (n+1) − (n+1))) ≤ 1 +
ln(n+1) for n::nat
  proof −
    have (∑ i=1..n+1. ln i) ≤ (∑ i=1..n. ln i) + ln (n+1)
      by simp
    also have (∑ i=1..n. ln i) ≤ (∑ i=1..n. ln (k i))
      by (intro sum-mono, subst ln-le-cancel-iff) (auto simp: Suc-le-eq dest: lb ub)
    also have ... = (∑ i=1..n. ln (k i) + 1) − n
      by (simp add: sum.distrib)
    also from * have ... = (n+1) * ln (n+1) − n
      by simp
    finally have a-minus-b: (∑ i=1..n+1. ln i) − ((n+1) * ln (n+1) − (n+1))
≤ 1 + ln (n+1)
      by simp

    from * have (n+1) * ln (n+1) − n = (∑ i=1..n. ln (k i) + 1) − n
      by simp
    also have ... = (∑ i=1..n. ln (k i))
      by (simp add: sum.distrib)
    also have ... ≤ (∑ i=1..n. ln (i+1))
      by (intro sum-mono, subst ln-le-cancel-iff) (auto simp: Suc-le-eq dest: lb ub)
   also from sum-shift-bounds-cl-nat-ivl[of ln 1 1 n] have ... = (∑ i=1+1..n+1.
ln i) ..
    also have ... = (∑ i=1..n+1. ln i)
      by (rule sum.mono-neutral-left) auto
    finally have b-minus-a: ((n+1) * ln (n+1) − (n+1)) − (∑ i=1..n+1. ln i)
≤ 1
      by simp
    have 0 ≤ ln (n+1)
      by simp
    with b-minus-a have ((n+1) * ln (n+1) − (n+1)) − (∑ i=1..n+1. ln i) ≤
1 + ln (n+1)
      by linarith
    with a-minus-b show ?thesis
      by linarith
  qed
  from ‹n > 0› have n ≥ 1 by simp
  thus ?thesis
  proof (induction n rule: dec-induct)
    case base
    then show ?case by simp
  next
    case (step n)
    from ln-fact[of n+1] **[of n] show ?case by simp
  qed
qed

lemma ln-fact-diff-bounds:
```

```
  abs(ln (fact n) − 2 ∗ ln (fact (n div 2)) − n ∗ ln 2) ≤ 4 ∗ ln (if n = 0 then 1
else n) + 3
proof (cases n div 2 = 0)
  case True
  hence n ≤ 1 by simp
  with ln-le-minus-one[of 2::real] show ?thesis by (cases n) simp-all
next
  case False
  then have n > 1 by simp
  let ?a = real n ∗ ln 2
  let ?b = 4 ∗ ln (real n) + 3
  let ?l1 = ln (fact (n div 2))
  let ?a1 = real (n div 2) ∗ ln (real (n div 2)) − real (n div 2)
  let ?b1 = 1 + ln (real (n div 2))
  let ?l2 = ln (fact n)
  let ?a2 = real n ∗ ln (real n) − real n
  let ?b2 = 1 + ln (real n)
  have abs-a: abs(?a − (?a2 − 2 ∗ ?a1)) ≤ ?b − 2 ∗ ?b1 − ?b2
  proof (cases even n)
    case True
    then have real (2 ∗ (n div 2)) = real n
      by simp
    then have n-div-2: real (n div 2) = real n / 2
      by simp
    from ‹n > 1› have ∗: abs(?a − (?a2 − 2 ∗ ?a1)) = 0
      by (simp add: n-div-2 ln-div algebra-simps)
    from ‹even n› and ‹n > 1› have 0 ≤ ln (real n) − ln (real (n div 2))
      by (auto elim: evenE)
    also have 2 ∗ ... ≤ 3 ∗ ln (real n) − 2 ∗ ln (real (n div 2))
      using ‹n > 1› by (auto intro!: ln-ge-zero)
    also have ... = ?b − 2 ∗ ?b1 − ?b2 by simp
    finally show ?thesis using ∗ by simp
  next
    case False
    then have real (2 ∗ (n div 2)) = real (n − 1)
      by simp
    with ‹n > 1› have n-div-2: real (n div 2) = (real n − 1) / 2
      by simp
    from ‹odd n› ‹n div 2 ≠ 0› have n ≥ 3
      by presburger

    have ?a − (?a2 − 2 ∗ ?a1) = real n ∗ ln 2 − real n ∗ ln (real n) + real n +
          2 ∗ real (n div 2) ∗ ln (real (n div 2)) − 2∗ real (n div 2)
      by (simp add: algebra-simps)
    also from n-div-2 have 2 ∗ real (n div 2) = real n − 1
      by simp
    also have real n ∗ ln 2 − real n ∗ ln (real n) + real n +
                (real n − 1) ∗ ln (real (n div 2)) − (real n − 1)
              = real n ∗ (ln (real n − 1) − ln (real n)) − ln (real (n div 2)) + 1
```

```
    using ‹n > 1› by (simp add: algebra-simps n-div-2 ln-div)
  finally have lhs: abs(?a − (?a2 − 2 ∗ ?a1)) =
      abs(real n ∗ (ln (real n − 1) − ln (real n)) − ln (real (n div 2)) + 1)
    by simp

  from ‹n > 1› have real n ∗ (ln (real n − 1) − ln (real n)) ≤ 0
    by (simp add: algebra-simps mult-left-mono)
  moreover from ‹n > 1› have ln (real (n div 2)) ≤ ln (real n) by simp
  moreover {
    have exp 1 ≤ (3::real) by (rule exp-le)
    also from ‹n ≥ 3› have ... ≤ exp (ln (real n)) by simp
    finally have ln (real n) ≥ 1 by simp
  }
  ultimately have ub: real n ∗ (ln (real n − 1) − ln (real n)) − ln(real (n div
2)) + 1 ≤
                        3 ∗ ln (real n) − 2 ∗ ln(real (n div 2)) by simp

  have mon: real n′ ∗ (ln (real n′) − ln (real n′ − 1)) ≤
              real n ∗ (ln (real n) − ln (real n − 1))
    if n ≥ 3 n′ ≥ n for n n′::nat
  proof (rule DERIV-nonpos-imp-nonincreasing[where f = λx. x ∗ (ln x − ln
(x − 1))], goal-cases)
    case 2
    show ?case
    proof clarify
      fix t assume t: real n ≤ t t ≤ real n′
      with that have 1 / (t − 1) ≥ ln (1 + 1/(t − 1))
        by (intro ln-add-one-self-le-self) simp-all
      also from t that have ln (1 + 1/(t − 1)) = ln t− ln (t − 1)
        by (simp add: ln-div [symmetric] field-simps)
      finally have ln t − ln (t − 1) ≤ 1 / (t − 1) .
      with that t
        show ∃ y. ((λx. x ∗ (ln x − ln (x − 1))) has-field-derivative y) (at t) ∧ y
≤ 0
        by (intro exI[of - 1 / (1 − t) + ln t − ln (t − 1)])
          (force intro!: derivative-eq-intros simp: field-simps)+
    qed
  qed (insert that, simp-all)

  from ‹n > 1› have ln 2 = ln (real n) − ln (real n / 2)
    by (simp add: ln-div)
  also from ‹n > 1› have ... ≤ ln (real n) − ln (real (n div 2))
    by simp
  finally have ∗: 3∗ln 2 + ln(real (n div 2)) ≤ 3∗ ln(real n) − 2∗ ln(real (n
div 2))
    by simp

  have − real n ∗ (ln (real n − 1) − ln (real n)) + ln(real (n div 2)) − 1 =
        real n ∗ (ln (real n) − ln (real n − 1)) − 1 + ln(real (n div 2))
```

```
    by (simp add: algebra-simps)
  also have real n * (ln (real n) - ln (real n - 1)) ≤ 3 * (ln 3 - ln (3 - 1))
    using mon[OF - ‹n ≥ 3›] by simp
  also {
    have Some (Float 3 (-1)) = ub-ln 1 3 by code-simp
    from ub-ln(1)[OF this] have ln 3 ≤ (1.6 :: real) by simp
    also have 1.6 - 1 / 3 ≤ 2 * (2/3 :: real) by simp
    also have 2/3 ≤ ln (2 :: real) by (rule ln-2-ge')
    finally have ln 3 - 1 / 3 ≤ 2 * ln (2 :: real) by simp
  }
  hence 3 * (ln 3 - ln (3 - 1)) - 1 ≤ 3 * ln (2 :: real) by simp
  also note *
  finally have - real n * (ln (real n - 1) - ln (real n)) + ln(real (n div 2)) -
1 ≤
                3 * ln (real n) - 2 * ln (real (n div 2)) by simp
  hence lhs': abs(real n * (ln (real n - 1) - ln (real n)) - ln(real (n div 2)) +
1) ≤
              3 * ln (real n) - 2 * ln (real (n div 2))
    using ub by simp
  have rhs: ?b - 2 * ?b1 - ?b2 = 3* ln (real n) - 2 * ln (real (n div 2))
    by simp
  from ‹n > 1› have ln (real (n div 2)) ≤ 3* ln (real n) - 2* ln (real (n div
2))
    by simp
  with rhs lhs lhs' show ?thesis
    by simp
qed
then have minus-a: -?a ≤ ?b - 2 * ?b1 - ?b2 - (?a2 - 2 * ?a1)
  by simp
from abs-a have a: ?a ≤ ?b - 2 * ?b1 - ?b2 + ?a2 - 2 * ?a1
  by (simp add: abs-if split: if-splits)
from ln-fact-bounds[of n div 2] False have abs-l1: abs(?l1 - ?a1) ≤ ?b1
  by (simp add: algebra-simps)
then have minus-l1: ?a1 - ?l1 ≤ ?b1
  by linarith
from abs-l1 have l1: ?l1 - ?a1 ≤ ?b1
  by linarith
from ln-fact-bounds[of n] False have abs-l2: abs(?l2 - ?a2) ≤ ?b2
  by (simp add: algebra-simps)
then have l2: ?l2 - ?a2 ≤ ?b2
  by simp
from abs-l2 have minus-l2: ?a2 - ?l2 ≤ ?b2
  by simp
from minus-a minus-l1 l2 have ?l2 - 2 * ?l1 - ?a ≤ ?b
  by simp
moreover from a l1 minus-l2 have - ?l2 + 2 * ?l1 + ?a ≤ ?b
  by simp
ultimately have abs((?l2 - 2*?l1) - ?a) ≤ ?b
  by simp
```

```
  then show ?thesis
    by simp
qed

lemma ln-primefact:
  assumes n ≠ 0
  shows   ln n = (∑ d=1..n. if primepow d ∧ d dvd n then ln (aprimedivisor d)
else 0)
          (is ?lhs = ?rhs)
proof −
  have ?rhs = (∑ d∈{x ∈ {1..n}. primepow x ∧ x dvd n}. ln (real (aprimedivisor
d)))
    unfolding primepows-def by (subst sum.inter-filter [symmetric]) simp-all
  also have {x ∈ {1..n}. primepow x ∧ x dvd n} = primepows n
   using assms by (auto simp: primepows-def dest: dvd-imp-le primepow-gt-Suc-0)
  finally have ∗: (∑ d∈primepows n. ln (real (aprimedivisor d))) = ?rhs ..
  from in-prime-factors-imp-prime prime-gt-0-nat
    have pf-pos: ⋀p. p∈#prime-factorization n ⟹ p > 0
    by blast
  from ln-msetprod[of prime-factorization n, OF pf-pos] assms
    have ln n = (∑ p∈#prime-factorization n. ln p)
      by (simp add: of-nat-prod-mset)
  also from ∗ sum-prime-factorization-conv-sum-primepows[of n ln, OF assms(1)]
    have ... = ?rhs by simp
  finally show ?thesis .
qed

context
begin

private lemma divisors:
  fixes x d::nat
  assumes x ∈ {1..n}
  assumes d dvd x
  shows ∃ k∈{1..n div d}. x = d ∗ k
proof −
  from assms have x ≤ n
    by simp
  then have ub: x div d ≤ n div d
    by (simp add: div-le-mono ‹x ≤ n›)
  from assms have 1 ≤ x div d by (auto elim!: dvdE)
  with ub have x div d ∈ {1..n div d}
    by simp
  with ‹d dvd x› show ?thesis by (auto intro!: bexI[of - x div d])
qed

lemma ln-fact-conv-mangoldt:
  shows ln (fact n) = (∑ d=1..n. mangoldt d ∗ floor (n / d))
proof −
```

```
  have *: (∑ da=1..n. if primepow da ∧
       da dvd d then ln (aprimedivisor da) else 0) =
       (∑ da=1..d. if primepow da ∧
       da dvd d then ln (aprimedivisor da) else 0) if d: d ∈ {1..n} for d
    by (rule sum.mono-neutral-right, insert d) (auto dest: dvd-imp-le)
  have (∑ d=1..n. ∑ da=1..d. if primepow da ∧
      da dvd d then ln (aprimedivisor da) else 0) =
      (∑ d=1..n. ∑ da=1..n. if primepow da ∧
      da dvd d then ln (aprimedivisor da) else 0)
    by (rule sum.cong) (insert *, simp-all)
  also have ... = (∑ da=1..n. ∑ d=1..n. if primepow da ∧
                  da dvd d then ln (aprimedivisor da) else 0)
    by (rule sum.commute)
  also have ... = sum (λd. mangoldt d * floor (n/d)) {1..n}
  proof (rule sum.cong)
    fix d assume d: d ∈ {1..n}
    have (∑ da = 1..n. if primepow d ∧ d dvd da then ln (real (aprimedivisor d))
else 0) =
          (∑ da = 1..n. if d dvd da then mangoldt d else 0)
      by (intro sum.cong) (simp-all add: mangoldt-def)
    also have ... = mangoldt d * real (card {x. x ∈ {1..n} ∧ d dvd x})
      by (subst sum.inter-filter [symmetric]) (simp-all add: algebra-simps)
    also {
      have {x. x ∈ {1..n} ∧ d dvd x} = {x. ∃k ∈{1..n div d}. x=k*d}
      proof safe
        fix x assume x ∈ {1..n} d dvd x
        thus ∃k∈{1..n div d}. x = k * d using divisors[of x n d] by auto
      next
        fix x k assume k: k ∈ {1..n div d}
        from k have k * d ≤ n div d * d by (intro mult-right-mono) simp-all
        also have n div d * d ≤ n div d * d + n mod d by (rule le-add1)
        also have ... = n by simp
        finally have k * d ≤ n .
        thus k * d ∈ {1..n} using d k by auto
      qed auto
      also have ... = (λk. k*d) ‘ {1..n div d}
        by fast
      also have card ... = card {1..n div d}
        by (rule card-image) (simp add: inj-on-def)
      also have ... = n div d
        by simp
      also have ... = ⌊n / d⌋
        by (simp add: floor-divide-of-nat-eq)
      finally have real (card {x. x ∈ {1..n} ∧ d dvd x}) = real-of-int ⌊n / d⌋
        by force
    }
    finally show (∑ da = 1..n. if primepow d ∧ d dvd da then ln (real (aprimedivisor
d)) else 0) =
          mangoldt d * real-of-int ⌊real n / real d⌋ .
```

```
  qed simp-all
  finally have (∑ d=1..n. ∑ da=1..d. if primepow da ∧
      da dvd d then ln (aprimedivisor da) else 0) =
    sum (λd. mangoldt d * floor (n/d)) {1..n} .
  with ln-primefact have (∑ d=1..n. ln d) =
    (∑ d=1..n. mangoldt d * floor (n/d))
    by simp
  with ln-fact show ?thesis
    by simp
qed

end

lemma mangoldt-pos: 0 ≤ mangoldt d
  using aprimedivisor-gt-1[of d]
 by (auto simp: mangoldt-def of-nat-le-iff[of 1 x for x, unfolded of-nat-1] Suc-le-eq
          intro!: ln-ge-zero dest: primepow-gt-Suc-0)

context
begin

private lemma div-2-mult-2-bds:
  fixes n d :: nat
  assumes d > 0
  shows 0 ≤ ⌊n / d⌋ − 2 * ⌊(n div 2) / d⌋ ⌊n / d⌋ − 2 * ⌊(n div 2) / d⌋ ≤ 1
proof −
  have ⌊2::real⌋ * ⌊(n div 2) / d⌋ ≤ ⌊2 * ((n div 2) / d)⌋
    by (rule le-mult-floor) simp-all
   also from assms have ... ≤ ⌊n / d⌋ by (intro floor-mono) (simp-all add:
field-simps)
  finally show 0 ≤ ⌊n / d⌋ − 2 * ⌊(n div 2) / d⌋ by (simp add: algebra-simps)
next
  have real (n div d) ≤ real (2 * ((n div 2) div d) + 1)
   by (subst div-mult2-eq [symmetric], simp only: mult.commute, subst div-mult2-eq)
simp
  thus ⌊n / d⌋ − 2 * ⌊(n div 2) / d⌋ ≤ 1
    unfolding of-nat-add of-nat-mult floor-conv-div-nat [symmetric] by simp-all
qed

private lemma n-div-d-eq-1: d ∈ {n div 2 + 1..n} ⟹ ⌊real n / real d⌋ = 1
  by (cases n = d) (auto simp: field-simps intro: floor-eq)

lemma psi-bounds-ln-fact:
  shows ln (fact n) − 2 * ln (fact (n div 2)) ≤ psi n
        psi n − psi (n div 2) ≤ ln (fact n) − 2 * ln (fact (n div 2))
proof −
  fix n::nat
  let ?k = n div 2 and ?d = n mod 2
  have *: ⌊?k / d⌋ = 0 if d > ?k for d
```

proof −
from *that div-less* **have** *0 = ?k div d* **by** *simp*
also have ... = ⌊*?k / d*⌋ **by** (*rule floor-divide-of-nat-eq* [*symmetric*])
finally show ⌊*?k / d*⌋ = *0* **by** *simp*
qed
have *sum-eq*: (∑ *d=1..2∗?k+?d. mangoldt d ∗* ⌊*?k / d*⌋) = (∑ *d=1..?k. mangoldt d ∗* ⌊*?k / d*⌋)
by (*intro sum.mono-neutral-right*) (*auto simp*: ∗)
from *ln-fact-conv-mangoldt* **have** *ln* (*fact n*) = (∑ *d=1..n. mangoldt d ∗* ⌊*n / d*⌋) **.**
also have ... = (∑ *d=1..n. mangoldt d ∗* ⌊(*2 ∗* (*n div 2*) + *n mod 2*) / *d*⌋)
by *simp*
also have ... ≤ (∑ *d=1..n. mangoldt d ∗* (*2 ∗* ⌊*?k / d*⌋ + *1*))
using *div-2-mult-2-bds*(*2*)[*of - n*]
by (*intro sum-mono mult-left-mono*, *subst of-int-le-iff*)
(*auto simp*: *algebra-simps mangoldt-pos*)
also have ... = *2 ∗* (∑ *d=1..n. mangoldt d ∗* ⌊(*n div 2*) / *d*⌋) + (∑ *d=1..n. mangoldt d*)
by (*simp add*: *algebra-simps sum.distrib sum-distrib-left*)
also have ... = *2 ∗* (∑ *d=1..2∗?k+?d. mangoldt d ∗* ⌊(*n div 2*) / *d*⌋) + (∑ *d=1..n. mangoldt d*)
by *presburger*
also from *sum-eq* **have** ... = *2 ∗* (∑ *d=1..?k. mangoldt d ∗* ⌊(*n div 2*) / *d*⌋) + (∑ *d=1..n. mangoldt d*)
by *presburger*
also from *ln-fact-conv-mangoldt psi-def* **have** ... = *2 ∗ ln* (*fact ?k*) + *psi n*
by *presburger*
finally show *ln* (*fact n*) − *2 ∗ ln* (*fact* (*n div 2*)) ≤ *psi n*
by *simp*
next
fix *n*::*nat*
let *?k = n div 2* **and** *?d = n mod 2*
from *psi-def* **have** *psi n − psi ?k* = (∑ *d=1..2∗?k+?d. mangoldt d*) − (∑ *d=1..?k. mangoldt d*)
by *presburger*
also have ... = *sum mangoldt* ({*1..2 ∗* (*n div 2*) + *n mod 2*} − {*1..n div 2*})
by (*subst sum-diff*) *simp-all*
also have ... = (∑ *d*∈({*1..2 ∗* (*n div 2*) + *n mod 2*} − {*1..n div 2*}).
(*if d* ≤ *?k then 0 else mangoldt d*))
by (*intro sum.cong*) *simp-all*
also have ... = (∑ *d=1..2∗?k+?d.* (*if d* ≤ *?k then 0 else mangoldt d*))
by (*intro sum.mono-neutral-left*) *auto*
also have ... = (∑ *d=1..n.* (*if d* ≤ *?k then 0 else mangoldt d*))
by *presburger*
also have ... = (∑ *d=1..n.* (*if d* ≤ *?k then mangoldt d ∗ 0 else mangoldt d*))
by (*intro sum.cong*) *simp-all*
also from *div-2-mult-2-bds*(*1*) **have** ... ≤ (∑ *d=1..n.* (*if d* ≤ *?k then mangoldt d ∗* (⌊*n*/*d*⌋ − *2 ∗* ⌊*?k*/*d*⌋) *else mangoldt d*))
by (*intro sum-mono*)

(*auto simp*: *algebra-simps mangoldt-pos intro*!: *mult-left-mono simp del*: *of-int-mult*)
also from *n-div-d-eq-1* **have** ... = ($\sum$ *d*=*1*..*n*. (*if d* ≤ *?k then mangoldt d* * (⌊*n*/*d*⌋ − *2* * ⌊*?k*/*d*⌋) *else mangoldt d* * ⌊*n*/*d*⌋))
by (*intro sum.cong refl*) *auto*
also have ... = ($\sum$ *d*=*1*..*n*. *mangoldt d* * *real-of-int* (⌊*real n* / *real d*⌋) −
(*if d* ≤ *?k then 2* * *mangoldt d* * *real-of-int* ⌊*real ?k* / *real d*⌋ *else 0*))
by (*intro sum.cong refl*) (*auto simp*: *algebra-simps*)
also have ... = ($\sum$ *d*=*1*..*n*. *mangoldt d* * *real-of-int* (⌊*real n* / *real d*⌋)) −
($\sum$ *d*=*1*..*n*. (*if d* ≤ *?k then 2* * *mangoldt d* * *real-of-int* ⌊*real ?k* / *real d*⌋ *else 0*))
by (*rule sum-subtractf*)
also have ($\sum$ *d*=*1*..*n*. (*if d* ≤ *?k then 2* * *mangoldt d* * *real-of-int* ⌊*real ?k* / *real d*⌋ *else 0*)) =
($\sum$ *d*=*1*..*?k*. (*if d* ≤ *?k then 2* * *mangoldt d* * *real-of-int* ⌊*real ?k* / *real d*⌋ *else 0*))
by (*intro sum.mono-neutral-right*) *auto*
also have ... = ($\sum$ *d*=*1*..*?k*. *2* * *mangoldt d* * *real-of-int* ⌊*real ?k* / *real d*⌋)
by (*intro sum.cong*) *simp-all*
also have ... = *2* * ($\sum$ *d*=*1*..*?k*. *mangoldt d* * *real-of-int* ⌊*real ?k* / *real d*⌋)
by (*simp add*: *sum-distrib-left mult-ac*)
also have ($\sum$ *d* = *1*..*n*. *mangoldt d* * *real-of-int* ⌊*real n* / *real d*⌋) − ... =
ln (*fact n*) − *2* * *ln* (*fact* (*n div 2*))
by (*simp add*: *ln-fact-conv-mangoldt*)
finally show *psi n* − *psi* (*n div 2*) ≤ *ln* (*fact n*) − *2* * *ln* (*fact* (*n div 2*)) .
qed

end

lemma *psi-bounds-induct*:
real n * *ln 2* − (*4* * *ln* (*real* (*if n* = *0 then 1 else n*)) + *3*) ≤ *psi n*
psi n − *psi* (*n div 2*) ≤ *real n* * *ln 2* + (*4* * *ln* (*real* (*if n* = *0 then 1 else n*)) + *3*)
proof −
from *le-imp-neg-le*[*OF ln-fact-diff-bounds*]
have *n* * *ln 2* − (*4* * *ln* (*if n* = *0 then 1 else n*) + *3*)
≤ *n* * *ln 2* − *abs*(*ln* (*fact n*) − *2* * *ln* (*fact* (*n div 2*)) − *n* * *ln 2*)
by *simp*
also have ... ≤ *ln* (*fact n*) − *2* * *ln* (*fact* (*n div 2*))
by *simp*
also from *psi-bounds-ln-fact* (*1*) **have** ... ≤ *psi n*
by *simp*
finally show *real n* * *ln 2* − (*4* * *ln* (*real* (*if n* = *0 then 1 else n*)) + *3*) ≤ *psi n* .
next
from *psi-bounds-ln-fact* (*2*) **have** *psi n* − *psi* (*n div 2*) ≤ *ln* (*fact n*) − *2* * *ln* (*fact* (*n div 2*)) .
also have ... ≤ *n* * *ln 2* + *abs*(*ln* (*fact n*) − *2* * *ln* (*fact* (*n div 2*)) − *n* * *ln*

2)
  by *simp*
 also from *ln-fact-diff-bounds* [*of n*]
  have *abs*(*ln* (*fact n*) − *2* ∗ *ln* (*fact* (*n div 2*)) − *n* ∗ *ln 2*)
      ≤ (*4* ∗ *ln* (*real* (*if n* = *0 then 1 else n*)) + *3*) **by** *simp*
 finally show *psi n* − *psi* (*n div 2*) ≤ *real n* ∗ *ln 2* + (*4* ∗ *ln* (*real* (*if n* = *0 then 1 else n*)) + *3*)
  by *simp*
qed

0.5 Bounding the psi function

In this section, we will first prove the relatively tight estimate *psi n* ≤ *3 / 2* + *ln 2* ∗ *real n* for *n* ≤ (*128*::′*a*) and then use the recurrence we have just derived to extend it to *psi n* ≤ *551 / 256* for *n* ≤ (*1024*::′*a*), at which point applying the recurrence can be used to prove the same bound for arbitrarily big numbers.

First of all, we will prove the bound for *n* ≤ (*128*::′*a*) using reflection and approximation.

context
begin

private lemma *Ball-insertD*:
 assumes ∀ *x*∈*insert y A*. *P x*
 shows *P y* ∀ *x*∈*A*. *P x*
 using *assms* **by** *auto*

private lemma *meta-eq-TrueE*: *PROP A* ≡ *Trueprop True* ⟹ *PROP A*
 by *simp*

private lemma *pre-mangoldt-pos*: *pre-mangoldt n* > *0*
 unfolding *pre-mangoldt-def* **by** (*auto simp*: *primepow-gt-Suc-0*)

private lemma *psi-conv-pre-mangoldt*: *psi n* = *ln* (*real* (*prod pre-mangoldt* {*1*..*n*}))
 by (*auto simp*: *psi-def mangoldt-def pre-mangoldt-def ln-prod primepow-gt-Suc-0 intro*!: *sum.cong*)

private lemma *eval-psi-aux1*: *psi 0* = *ln* (*real* (*numeral Num.One*))
 by (*simp add*: *psi-def*)

private lemma *eval-psi-aux2*:
 assumes *psi m* = *ln* (*real* (*numeral x*)) *pre-mangoldt n* = *y m* + *1* = *n numeral x* ∗ *y* = *z*
 shows *psi n* = *ln* (*real z*)
proof −
 from *assms*(*2*) [*symmetric*] **have** [*simp*]: *y* > *0* **by** (*simp add*: *pre-mangoldt-pos*)
 have *psi n* = *psi* (*Suc m*) **by** (*simp add*: *assms*(*3*) [*symmetric*])
 also have ... = *ln* (*real y* ∗ (∏ *x* = *Suc 0*..*m*. *real* (*pre-mangoldt x*)))

using *assms*(*2*,*3*) [*symmetric*] **by** (*simp add*: *psi-conv-pre-mangoldt prod-nat-ivl-Suc' mult-ac*)
also have ... = *ln* (*real y*) + *psi m*
by (*subst ln-mult*) (*simp-all add*: *pre-mangoldt-pos prod-pos psi-conv-pre-mangoldt*)
also have *psi m* = *ln* (*real* (*numeral x*)) **by** *fact*
also have *ln* (*real y*) + ... = *ln* (*real* (*numeral x* * *y*)) **by** (*simp add*: *ln-mult*)
finally show *?thesis* **by** (*simp add*: *assms*(*4*) [*symmetric*])
qed

private lemma *Ball-atLeast0AtMost-doubleton*:
assumes *psi 0* ≤ *3 / 2* * *ln 2* * *real 0*
assumes *psi 1* ≤ *3 / 2* * *ln 2* * *real 1*
shows (∀*x*∈{*0..1*}. *psi x* ≤ *3 / 2* * *ln 2* * *real x*)
using *assms* **unfolding** *One-nat-def atLeast0-atMost-Suc ball-simps* **by** *auto*

private lemma *Ball-atLeast0AtMost-insert*:
assumes (∀*x*∈{*0..m*}. *psi x* ≤ *3 / 2* * *ln 2* * *real x*)
assumes *psi* (*numeral n*) ≤ *3 / 2* * *ln 2* * *real* (*numeral n*) *m* = *pred-numeral n*
shows (∀*x*∈{*0..numeral n*}. *psi x* ≤ *3 / 2* * *ln 2* * *real x*)
using *assms*
by (*subst numeral-eq-Suc*[*of n*], *subst atLeast0-atMost-Suc*,
subst ball-simps, *simp only*: *numeral-eq-Suc* [*symmetric*])

private lemma *eval-psi-ineq-aux*:
assumes *psi n* = *x x* ≤ *3 / 2* * *ln 2* * *n*
shows *psi n* ≤ *3 / 2* * *ln 2* * *n*
using *assms* **by** *simp-all*

private lemma *eval-psi-ineq-aux2*:
assumes *numeral m* ^ *2* ≤ (*2*::*nat*) ^ (*3* * *n*)
shows *ln* (*real* (*numeral m*)) ≤ *3 / 2* * *ln 2* * *real n*
proof −
have *ln* (*real* (*numeral m*)) ≤ *3 / 2* * *ln 2* * *real n* ⟷
2 * *log 2* (*real* (*numeral m*)) ≤ *3* * *real n*
by (*simp add*: *field-simps log-def*)
also have *2* * *log 2* (*real* (*numeral m*)) = *log 2* (*real* (*numeral m* ^ *2*))
by (*subst of-nat-power*, *subst log-nat-power*) *simp-all*
also have ... ≤ *3* * *real n* ⟷ *real* ((*numeral m*) ^ *2*) ≤ *2 powr real* (*3* * *n*)
by (*subst Transcendental.log-le-iff*) *simp-all*
also have *2 powr* (*3* * *n*) = *real* (*2* ^ (*3* * *n*))
by (*simp add*: *powr-realpow* [*symmetric*])
also have *real* ((*numeral m*) ^ *2*) ≤ ... ⟷ *numeral m* ^ *2* ≤ (*2*::*nat*) ^ (*3* * *n*)
by (*rule of-nat-le-iff*)
finally show *?thesis* **using** *assms* **by** *blast*
qed

private lemma *eval-psi-ineq-aux-mono*:

assumes *psi n = x psi m = x psi n ≤ 3 / 2 * ln 2 * n n ≤ m*
shows *psi m ≤ 3 / 2 * ln 2 * m*
proof −
from *assms* **have** *psi m = psi n* **by** *simp*
also have ... *≤ 3 / 2 * ln 2 * n* **by** *fact*
also from ‹*n ≤ m*› **have** ... *≤ 3 / 2 * ln 2 * m* **by** *simp*
finally show *?thesis* **.**
qed

ML-file ‹*bertrand.ML*›

local-setup ‹ *fn ctxt =>*
let
fun tac {context = ctxt, ...} =
let
val psi-cache = Bertrand.prove-psi ctxt 129
fun prove-psi-ineqs ctxt =
let
fun tac {context = ctxt, ...} =
HEADGOAL (resolve-tac ctxt @{thms eval-psi-ineq-aux2} THEN' Simplifier.simp-tac ctxt)
fun prove-by-approx n thm =
let
val thm = thm RS @{thm eval-psi-ineq-aux}
val [prem] = Thm.prems-of thm
val prem = Goal.prove ctxt [] [] prem tac
in
prem RS thm
end
fun prove-by-mono last-thm last-thm' thm =
let
val thm = @{thm eval-psi-ineq-aux-mono} OF [last-thm, thm, last-thm']
val [prem] = Thm.prems-of thm
val prem = Goal.prove ctxt [] [] prem (K (HEADGOAL (Simplifier.simp-tac ctxt)))
in
prem RS thm
end
fun go - acc [] = acc
| go last acc ((n, x, thm) :: xs) =
let
val thm' =
case last of
NONE => prove-by-approx n thm
| SOME (last-x, last-thm, last-thm') =>
if last-x = x then
prove-by-mono last-thm last-thm' thm

```
                    else
                      prove-by-approx n thm
                in
                  go (SOME (x, thm, thm')) (thm' :: acc) xs
                end
        in
          rev o go NONE []
        end

      val psi-ineqs = prove-psi-ineqs ctxt psi-cache
      fun prove-ball ctxt (thm1 :: thm2 :: thms) =
            let
              val thm = @{thm Ball-atLeast0AtMost-doubleton} OF [thm1, thm2]
              fun solve-prem thm =
                let
                    fun tac {context = ctxt, ...} = HEADGOAL (Simplifier.simp-tac
ctxt)
                val thm' = Goal.prove ctxt [] [] (Thm.cprem-of thm 1 |> Thm.term-of)
tac
                in
                  thm' RS thm
                end
                fun go thm thm' = (@{thm Ball-atLeast0AtMost-insert} OF [thm',
thm]) |> solve-prem
            in
              fold go thms thm
            end
        | prove-ball - - = raise Match
    in
      HEADGOAL (resolve-tac ctxt [prove-ball ctxt psi-ineqs])
    end
  val thm = Goal.prove @{context} [] [] @{prop ∀n∈{0..128}. psi n ≤ 3 / 2 * ln
2 * n} tac
in
  Local-Theory.note ((@{binding psi-ubound-log-128}, []), [thm]) ctxt |> snd
end
```

end

context
begin

private lemma *psi-ubound-aux*:
  defines $f \equiv \lambda x{::}real.\ (4 * ln\ x + 3)\ /\ (ln\ 2 * x)$
  assumes $x \geq 2\ x \leq y$
  shows $f\ x \geq f\ y$
using *assms*(3)

proof (*rule DERIV-nonpos-imp-nonincreasing*, *clarify*, *goal-cases*)
case (*1 t*)
define f' **where** $f' = (\lambda x.\ (1 - 4 * ln\ x)\ /\ x\hat{}2\ /\ ln\ 2 :: real)$
from *1 assms*(*2*) **have** (*f has-real-derivative f' t*) (*at t*) **unfolding** *f-def f'-def*
by (*auto intro*!: *derivative-eq-intros simp*: *field-simps power2-eq-square*)
moreover {
from *ln-2-ge* **have** $1/4 \le ln\ (2::real)$ **by** *simp*
also from *assms*(*2*) *1* **have** $\ldots \le ln\ t$ **by** *simp*
finally have $ln\ t \ge 1/4$ **.**
}
with *1 assms*(*2*) **have** $f'\ t \le 0$ **by** (*simp add*: *f'-def field-simps*)
ultimately show *?case* **by** (*intro exI*[*of* - $f'\ t$]) *simp-all*
qed

These next rules are used in combination with $real\ ?n * ln\ 2 - (4 * ln\ (real\ (if\ ?n = 0\ then\ 1\ else\ ?n)) + 3) \le psi\ ?n$ $psi\ ?n - psi\ (?n\ div\ 2) \le real\ ?n * ln\ 2 + (4 * ln\ (real\ (if\ ?n = 0\ then\ 1\ else\ ?n)) + 3)$ and $\forall n{\in}\{0..128\}.\ psi\ n \le 3\ /\ 2 * ln\ 2 * real\ n$ to extend the upper bound for *psi* from values no greater than 128 to values no greater than 1024. The constant factor of the upper bound changes every time, but once we have reached 1024, the recurrence is self-sustaining in the sense that we do not have to adjust the constant factor anymore in order to double the range.

lemma *psi-ubound-log-double-cases'*:
assumes $\bigwedge n.\ n \le m \Longrightarrow psi\ n \le c * ln\ 2 * real\ n\ n \le m'\ m' = 2{*}m$
$c \le c'\ c \ge 0\ m \ge 1\ c' \ge 1 + c/2 + (4 * ln\ (m{+}1) + 3)\ /\ (ln\ 2 * (m{+}1))$
shows $psi\ n \le c' * ln\ 2 * real\ n$
proof (*cases* $n > m$)
case *False*
hence $psi\ n \le c * ln\ 2 * real\ n$ **by** (*intro assms*) *simp-all*
also have $c \le c'$ **by** *fact*
finally show *?thesis* **by** − (*simp-all add*: *mult-right-mono*)
next
case *True*
hence *n*: $n \ge m{+}1$ **by** *simp*
from *psi-bounds-induct*(*2*)[*of n*] *True*
have $psi\ n \le real\ n * ln\ 2 + 4 * ln\ (real\ n) + 3 + psi\ (n\ div\ 2)$ **by** *simp*
also from *assms* **have** $psi\ (n\ div\ 2) \le c * ln\ 2 * real\ (n\ div\ 2)$
by (*intro assms*) *simp-all*
also have $real\ (n\ div\ 2) \le real\ n\ /\ 2$ **by** *simp*
also have $c * ln\ 2 * \ldots = c\ /\ 2 * ln\ 2 * real\ n$ **by** *simp*
also have $real\ n * ln\ 2 + 4 * ln\ (real\ n) + 3 + \ldots =$
$(1 + c/2) * ln\ 2 * real\ n + (4 * ln\ (real\ n) + 3)$ **by** (*simp add*: *field-simps*)
also {
have $(4 * ln\ (real\ n) + 3)\ /\ (ln\ 2 * (real\ n)) \le (4 * ln\ (m{+}1) + 3)\ /\ (ln\ 2 * (m{+}1))$

```
    using n assms by (intro psi-ubound-aux) simp-all
  also from assms have (4 * ln (m+1) + 3) / (ln 2 * (m+1)) ≤ c' - 1 - c/2

    by (simp add: algebra-simps)
  finally have 4 * ln (real n) + 3 ≤ (c' - 1 - c/2) * ln 2 * real n
    using n by (simp add: field-simps)
 }
 also have (1 + c / 2) * ln 2 * real n + (c' - 1 - c / 2) * ln 2 * real n = c'
* ln 2 * real n
  by (simp add: field-simps)
 finally show ?thesis using ‹c ≥ 0› by (simp-all add: mult-left-mono)
qed

end

lemma psi-ubound-log-double-cases:
 assumes ∀ n≤m. psi n ≤ c * ln 2 * real n
       c' ≥ 1 + c/2 + (4 * ln (m+1) + 3) / (ln 2 * (m+1))
       m' = 2*m c ≤ c' c ≥ 0 m ≥ 1
 shows  ∀ n≤m'. psi n ≤ c' * ln 2 * real n
 using assms(1) by (intro allI impI assms psi-ubound-log-double-cases'[of m c -
m' c']) auto

lemma psi-ubound-log-1024:
 ∀ n≤1024. psi n ≤ 551 / 256 * ln 2 * real n
proof −
 from psi-ubound-log-128 have ∀ n≤128. psi n ≤ 3 / 2 * ln 2 * real n by simp
 hence ∀ n≤256. psi n ≤ 1025 / 512 * ln 2 * real n
 proof (rule psi-ubound-log-double-cases, goal-cases)
  case 1
  have Some (Float 624 (− 7)) = ub-ln 9 129 by code-simp
  from ub-ln(1)[OF this] and ln-2-ge show ?case by (simp add: field-simps)
 qed simp-all
 hence ∀ n≤512. psi n ≤ 549 / 256 * ln 2 * real n
 proof (rule psi-ubound-log-double-cases, goal-cases)
  case 1
  have Some (Float 180 (− 5)) = ub-ln 7 257 by code-simp
  from ub-ln(1)[OF this] and ln-2-ge show ?case by (simp add: field-simps)
 qed simp-all
 thus ∀ n≤1024. psi n ≤ 551 / 256 * ln 2 * real n
 proof (rule psi-ubound-log-double-cases, goal-cases)
  case 1
  have Some (Float 203 (− 5)) = ub-ln 7 513 by code-simp
  from ub-ln(1)[OF this] and ln-2-ge show ?case by (simp add: field-simps)
 qed simp-all
qed

lemma psi-bounds-sustained-induct:
 assumes 4 * ln (1 + 2 ^ j) + 3 ≤ d * ln 2 * (1 + 2^j)
```

```
  assumes 4 / (1 + 2^j) ≤ d * ln 2
  assumes 0 ≤ c
  assumes c / 2 + d + 1 ≤ c
  assumes j ≤ k
  assumes ⋀n. n ≤ 2^k ⟹ psi n ≤ c * ln 2 * n
  assumes n ≤ 2^(Suc k)
  shows psi n ≤ c * ln 2 * n
proof (cases n ≤ 2^k)
  case True
  with assms(6) show ?thesis .
next
  case False
  from psi-bounds-induct(2)
    have psi n - psi (n div 2) ≤ real n * ln 2 + (4 * ln (real (if n = 0 then 1
else n)) + 3) .
  also from False have (if n = 0 then 1 else n) = n
    by simp
  finally have psi n ≤ real n * ln 2 + (4 * ln (real n) + 3) + psi (n div 2)
    by simp
  also from assms(6,7) have psi (n div 2) ≤ c * ln 2 * (n div 2)
    by simp
  also have real (n div 2) ≤ real n / 2
    by simp
  also have real n * ln 2 + (4 * ln (real n) + 3) + c * ln 2 * (n / 2) ≤ c * ln 2
* real n
    proof (rule overpower-lemma[of
          λx. x * ln 2 + (4 * ln x + 3) + c * ln 2 * (x / 2) 1+2^j
          λx. c * ln 2 * x λx. c * ln 2 - ln 2 - 4 / x - c / 2 * ln 2
          real n])
      from assms(1) have 4 * ln (1 + 2^j) + 3 ≤ d * ln 2 * (1 + 2^j) .
      also from assms(4) have d ≤ c - c/2 - 1
        by simp
      also have (...) * ln 2 * (1 + 2 ^ j) = c * ln 2 * (1 + 2 ^ j) - c / 2 * ln
2 * (1 + 2 ^ j)
          - (1 + 2 ^ j) * ln 2
        by (simp add: left-diff-distrib)
      finally have 4 * ln (1 + 2^j) + 3 ≤ c * ln 2 * (1 + 2 ^ j) - c / 2 * ln 2
* (1 + 2 ^ j)
          - (1 + 2 ^ j) * ln 2
        by (simp add: add-pos-pos)
      then show (1 + 2 ^ j) * ln 2 + (4 * ln (1 + 2 ^ j) + 3)
                  + c * ln 2 * ((1 + 2 ^ j) / 2) ≤ c * ln 2 * (1 + 2 ^ j)
        by simp
    next
      fix x::real
      assume x: 1 + 2^j ≤ x
      moreover have 1 + 2 ^ j > (0::real) by (simp add: add-pos-pos)
      ultimately have x-pos: x > 0 by linarith
      show ((λx. c * ln 2 * x - (x * ln 2 + (4 * ln x + 3) + c * ln 2 * (x / 2)))
```

```
        has-real-derivative c * ln 2 - ln 2 - 4 / x - c / 2 * ln 2) (at x)
      by (rule derivative-eq-intros refl | simp add: ‹0 < x›)+
    from ‹0 < x› ‹0 < 1 + 2^j› have 0 < x * (1 + 2^j)
      by (rule mult-pos-pos)
    have 4 / x ≤ 4 / (1 + 2^j)
      by (intro divide-left-mono mult-pos-pos add-pos-pos x x-pos) simp-all
    also from assms(2) have 4 / (1 + 2^j) ≤ d * ln 2 .
    also from assms(4) have d ≤ c - c/2 - 1 by simp
      also have ... * ln 2 = c * ln 2 - c/2 * ln 2 - ln 2 by (simp add:
algebra-simps)
    finally show 0 ≤ c * ln 2 - ln 2 - 4 / x - c / 2 * ln 2 by simp
  next
    have 1 + 2^j = real (1 + 2^j) by simp
    also from assms(5) have ... ≤ real (1 + 2^k) by simp
    also from False have 2^k ≤ n - 1 by simp
    finally show 1 + 2^j ≤ real n using False by simp
  qed
  finally show ?thesis using assms by - (simp-all add: mult-left-mono)
qed

lemma psi-bounds-sustained:
  assumes ⋀n. n ≤ 2^k ⟹ psi n ≤ c * ln 2 * n
  assumes 4 * ln (1 + 2^k) + 3 ≤ (c/2 - 1) * ln 2 * (1 + 2^k)
  assumes 4 / (1 + 2^k) ≤ (c/2 - 1) * ln 2
  assumes c ≥ 0
  shows psi n ≤ c * ln 2 * n
proof -
  have *: psi n ≤ c * ln 2 * n if n ≤ 2^j for j n
  using that
  proof (induction j arbitrary: n)
    case 0
     with assms(4) 0 show ?case unfolding psi-def mangoldt-def by (cases n)
auto
  next
    case (Suc j)
    show ?case
      proof (cases k ≤ j)
        case True
        from assms(4) have c-div-2: c/2 + (c/2 - 1) + 1 ≤ c
          by simp
        from psi-bounds-sustained-induct[of k c/2 -1 c j,
            OF assms(2) assms(3) assms(4) c-div-2 True Suc.IH Suc.prems]
          show ?thesis by simp
      next
        case False
        then have j-lt-k: Suc j ≤ k by simp
        from Suc.prems have n ≤ 2 ^ Suc j .
        also have (2::nat) ^ Suc j ≤ 2 ^ k
          using power-increasing[of Suc j k 2::nat, OF j-lt-k]
```

```
      by simp
    finally show ?thesis using assms(1) by simp
  qed
 qed
 have n < 2 ^ n by (induction n) simp-all
 with *[of n n] show ?thesis by simp
qed

lemma psi-ubound-log: psi n ≤ 551 / 256 * ln 2 * n
proof (rule psi-bounds-sustained)
 show 0 ≤ 551 / (256 :: real) by simp
next
 fix n :: nat assume n ≤ 2 ^ 10
 with psi-ubound-log-1024 show psi n ≤ 551 / 256 * ln 2 * real n by auto
next
 have 4 / (1 + 2 ^ 10) ≤ (551 / 256 / 2 - 1) * (2/3 :: real)
  by simp
 also have ... ≤ (551 / 256 / 2 - 1) * ln 2
  by (intro mult-left-mono ln-2-ge') simp-all
 finally show 4 / (1 + 2 ^ 10) ≤ (551 / 256 / 2 - 1) * ln (2 :: real) .
next
 have Some (Float 16 (-1)) = ub-ln 3 1025 by code-simp
 from ub-ln(1)[OF this] and ln-2-ge
  have 2048 * ln 1025 + 1536 ≤ 39975 * (ln 2::real) by simp
 thus 4 * ln (1 + 2 ^ 10) + 3 ≤ (551 / 256 / 2 - 1) * ln 2 * (1 + 2 ^ 10 ::
real)
  by simp
qed

lemma psi-ubound-3-2: psi n ≤ 3/2 * n
proof -
 have (551 / 256) * ln 2 ≤ (551 / 256) * (16/23 :: real)
  by (intro mult-left-mono ln-2-le') auto
 also have ... ≤ 3 / 2 by simp
 finally have 551 / 256 * ln 2 ≤ 3/(2::real) .
 with of-nat-0-le-iff mult-right-mono have 551 / 256 * ln 2 * n ≤ 3/2 * n
  by blast
 with psi-ubound-log[of n] show ?thesis
  by linarith
qed
```

0.6 Doubling psi and theta

```
lemma psi-residues-compare-2:
 psi-odd-2 n ≤ psi-even-2 n
proof -
 have psi-odd-2 n = (∑ d∈{d. d ∈ {2..n} ∧ primepow-odd d}. mangoldt-odd d)
  unfolding mangoldt-odd-def by (rule sum.mono-neutral-right) auto
 also have ... = (∑ d∈{d. d ∈ {2..n} ∧ primepow-odd d}. ln (real (aprimedivisor
```

```
d)))
    by (intro sum.cong refl) (simp add: mangoldt-odd-def)
  also have ... ≤ (∑ d∈{d. d ∈ {2..n} ∧ primepow-even d}. ln (real (aprimedivisor
d)))
   proof (rule sum-le-included [where i = λy. y * aprimedivisor y]; clarify?)
    fix d :: nat assume d ∈ {2..n} primepow-odd d
    note d = this
    then obtain p k where d': k ≥ 1 prime p d = p ^ (2*k+1)
      by (auto simp: primepow-odd-def)
    from d' have p ^ (2 * k) ≤ p ^ (2 * k + 1)
      by (subst power-increasing-iff) (auto simp: prime-gt-Suc-0-nat)
    also from d d' have ... ≤ n by simp
    finally have p ^ (2 * k) ≤ n .
    moreover from d' have p ^ (2 * k) > 1
      by (intro one-less-power) (simp-all add: prime-gt-Suc-0-nat)
    ultimately have p ^ (2 * k) ∈ {2..n} by simp
    moreover from d' have primepow-even (p ^ (2 * k))
      by (auto simp: primepow-even-def)
    ultimately show ∃ y∈{d ∈ {2..n}. primepow-even d}. y * aprimedivisor y =
d ∧
                    ln (real (aprimedivisor d)) ≤ ln (real (aprimedivisor y)) using
d'
      by (intro bexI[of - p ^ (2 * k)])
        (auto simp: aprimedivisor-prime-power aprimedivisor-primepow)
  qed (simp-all add: of-nat-ge-1-iff Suc-le-eq)
  also have ... = (∑ d∈{d. d ∈ {2..n} ∧ primepow-even d}. mangoldt-even d)
    by (intro sum.cong refl) (simp add: mangoldt-even-def)
  also have ... = psi-even-2 n
    unfolding mangoldt-even-def by (rule sum.mono-neutral-left) auto
  finally show ?thesis .
qed

lemma psi-residues-compare:
  psi-odd n ≤ psi-even n
proof −
  have ¬ primepow-odd 1 by (simp add: primepow-odd-def)
  hence *: mangoldt-odd 1 = 0 by (simp add: mangoldt-odd-def)
  have ¬ primepow-even 1
    using primepow-gt-Suc-0[OF primepow-even-imp-primepow, of 1] by auto
  with mangoldt-even-def have **: mangoldt-even 1 = 0
    by simp
  from psi-odd-def have psi-odd n = (∑ d=1..n. mangoldt-odd d)
    by simp
  also from * have ... = psi-odd-2 n
    by (cases n ≥ 1) (simp-all add: eval-nat-numeral sum-head-Suc)
  also from psi-residues-compare-2 have ... ≤ psi-even-2 n .
  also from ** have ... = psi-even n
    by (cases n ≥ 1) (simp-all add: eval-nat-numeral sum-head-Suc psi-even-def)
  finally show ?thesis .
```

qed

lemma *primepow-iff-even-sqr*:
primepow n $\longleftrightarrow$ *primepow-even* (*n* ^ *2*)
by (*auto simp*: *primepow-even-altdef aprimedivisor-primepow-power primepow-power-iff prime-elem-multiplicity-power-distrib prime-aprimedivisor prime-imp-prime-elem* *dest*: *primepow-gt-Suc-0*)

lemma *psi-sqrt*: *psi* (*Discrete.sqrt n*) = *psi-even n*
proof (*induction n*)
case *0*
with *psi-def psi-even-def* **show** *?case* **by** *simp*
next
case (*Suc n*)
then show *?case*
proof *cases*
assume *asm*: ∃ *m*. *Suc n* = *m* ^ *2*
with *sqrt-Suc* **have** *sqrt-seq*: *Discrete.sqrt*(*Suc n*) = *Suc*(*Discrete.sqrt n*)
by *simp*
from *asm* **obtain** *m* **where** *Suc n* = *m* ^ *2*
by *blast*
with *sqrt-seq* **have** *Suc*(*Discrete.sqrt n*) = *m*
by *simp*
with ‹*Suc n* = *m* ^ *2*› **have** *suc-sqrt-n-sqrt*: (*Suc*(*Discrete.sqrt n*)) ^ *2* = *Suc n*
by *simp*
from *sqrt-seq* **have** *psi* (*Discrete.sqrt* (*Suc n*)) = *psi* (*Suc* (*Discrete.sqrt n*))
by *simp*
also from *psi-def* **have** ... = *psi* (*Discrete.sqrt n*) + *mangoldt* (*Suc* (*Discrete.sqrt n*))
by *simp*
also from *Suc.IH* **have** *psi* (*Discrete.sqrt n*) = *psi-even n* .
also have *mangoldt* (*Suc* (*Discrete.sqrt n*)) = *mangoldt-even* (*Suc n*)
proof (*cases primepow* (*Suc*(*Discrete.sqrt n*)))
case *True*
with *primepow-iff-even-sqr* **have** *True2*: *primepow-even* ((*Suc*(*Discrete.sqrt n*)) ^ *2*)
by *simp*
from *suc-sqrt-n-sqrt* **have** *mangoldt-even* (*Suc n*) = *mangoldt-even* ((*Suc*(*Discrete.sqrt n*)) ^ *2*)
by *simp*
also from *mangoldt-even-def True2*
have ... = *ln* (*aprimedivisor* ((*Suc* (*Discrete.sqrt n*)) ^ *2*))
by *simp*
also from *True* **have** *aprimedivisor* ((*Suc* (*Discrete.sqrt n*)) ^ *2*) = *aprimedivisor* (*Suc* (*Discrete.sqrt n*))
by (*simp add*: *aprimedivisor-primepow-power*)
also from *True mangoldt-def*
have *ln* (...) = *mangoldt* (*Suc* (*Discrete.sqrt n*))
by *simp*

```
      finally show ?thesis ..
    next
      case False
      with primepow-iff-even-sqr
        have False2: ¬ primepow-even ((Suc(Discrete.sqrt n)) ^2)
        by simp
    from suc-sqrt-n-sqrt have mangoldt-even (Suc n) = mangoldt-even ((Suc(Discrete.sqrt
n)) ^2)
        by simp
      also from mangoldt-even-def False2
        have ... = 0
        by simp
      also from False mangoldt-def
        have ... = mangoldt (Suc (Discrete.sqrt n))
        by simp
      finally show ?thesis ..
    qed
   also from psi-even-def have psi-even n + mangoldt-even (Suc n) = psi-even
(Suc n)
      by simp
    finally show ?case .
  next
    assume asm: ¬(∃ m. Suc n = m ^2)
    with sqrt-Suc have sqrt-eq: Discrete.sqrt (Suc n) = Discrete.sqrt n
      by simp
    then have lhs: psi (Discrete.sqrt (Suc n)) = psi (Discrete.sqrt n)
      by simp
    have ¬ primepow-even (Suc n)
      proof
        assume primepow-even (Suc n)
        with primepow-even-def obtain p k
          where 1 ≤ k ∧ prime p ∧ Suc n = p ^ (2 * k)
          by blast
        with power-even-eq have Suc n = (p ^ k) ^2
          by simp
        with asm show False by blast
      qed
    with psi-even-def mangoldt-even-def
      have rhs: psi-even (Suc n) = psi-even n
      by simp
    from Suc.IH lhs rhs show ?case
      by simp
  qed
qed

lemma mangoldt-split:
  mangoldt d = mangoldt-1 d + mangoldt-even d + mangoldt-odd d
proof (cases primepow d)
  case False
```

thus *?thesis*
by (*auto simp*: *mangoldt-def mangoldt-1-def mangoldt-even-def mangoldt-odd-def*
dest: *primepow-even-imp-primepow primepow-odd-imp-primepow*)
next
case *True*
thus *?thesis*
by (*auto simp*: *mangoldt-def mangoldt-1-def mangoldt-even-def mangoldt-odd-def primepow-cases*)
qed

lemma *psi-split*: *psi n = theta n + psi-even n + psi-odd n*
by (*induction n*)
(*simp-all add*: *psi-def theta-def psi-even-def psi-odd-def mangoldt-1-def mangoldt-split*)

lemma *psi-mono*: $m \leq n \Longrightarrow$ *psi m* $\leq$ *psi n*
using *mangoldt-pos sum-mono2*[*of* $\{1..n\}$ $\{1..m\}$ *mangoldt*] **by** (*simp add*: *psi-def*)

lemma *psi-pos*: *0* $\leq$ *psi n*
by (*auto simp*: *psi-def intro*!: *sum-nonneg mangoldt-pos*)

lemma *mangoldt-odd-pos*: *0* $\leq$ *mangoldt-odd d*
using *aprimedivisor-gt-Suc-0*[*of d*]
by (*auto simp*: *mangoldt-odd-def of-nat-le-iff*[*of 1*, *unfolded of-nat-1*]
simp del: *aprimedivisor-gt-Suc-0 intro*!: *ln-ge-zero*
dest!: *primepow-odd-imp-primepow primepow-gt-Suc-0*)

lemma *psi-odd-mono*: $m \leq n \Longrightarrow$ *psi-odd m* $\leq$ *psi-odd n*
using *mangoldt-odd-pos sum-mono2*[*of* $\{1..n\}$ $\{1..m\}$ *mangoldt-odd*]
by (*simp add*: *psi-odd-def*)

lemma *psi-odd-pos*: *0* $\leq$ *psi-odd n*
by (*auto simp*: *psi-odd-def intro*!: *sum-nonneg mangoldt-odd-pos*)

lemma *psi-theta*:
theta n + psi (*Discrete.sqrt n*) $\leq$ *psi n psi n* $\leq$ *theta n + 2 * psi* (*Discrete.sqrt n*)
using *psi-odd-pos*[*of n*] *psi-residues-compare*[*of n*] *psi-sqrt*[*of n*] *psi-split*[*of n*]
by *simp-all*

context
begin

private lemma *sum-minus-one*:
$(\sum x \in \{1..y\}.\ (-\ 1 :: real)\ \hat{}\ (x + 1)) =$ (*if odd y then 1 else 0*)
by (*induction y*) *simp-all*

private lemma *div-invert*:
fixes *x y n* :: *nat*

assumes $x > 0$ $y > 0$ $y \leq n$ *div* x
shows $x \leq n$ *div* y
proof −
from *assms*(*1*,*3*) **have** $y * x \leq (n$ *div* $x) * x$
by *simp*
also have ... $\leq n$
by (*simp add*: *minus-mod-eq-div-mult*[*symmetric*])
finally have $y * x \leq n$ **.**
with *assms*(*2*) **show** *?thesis*
using *div-le-mono*[*of* $y*x$ n y] **by** *simp*
qed

lemma *sum-expand-lemma*:
$(\sum d{=}1..n.\ (-1)\ \hat{}\ (d + 1) * psi\ (n$ *div* $d)) =$
$(\sum d = 1..n.$ (*if odd* $(n$ *div* $d)$ *then 1 else 0*$) * mangoldt\ d)$
proof −
have $**$: $x \leq n$ **if** $x \leq n$ *div* y **for** x y
using *div-le-dividend order-trans that* **by** *blast*
have $(\sum d{=}1..n.\ (-1)\hat{}(d{+}1) * psi\ (n$ *div* $d)) =$
$(\sum d{=}1..n.\ (-1)\hat{}(d{+}1) * (\sum e{=}1..n$ *div* $d.\ mangoldt\ e))$
by (*simp add*: *psi-def*)
also have ... $= (\sum d = 1..n. \sum e = 1..n$ *div* $d.\ (-1)\hat{}(d{+}1) * mangoldt\ e)$
by (*simp add*: *sum-distrib-left*)
also from $**$ **have** ... $= (\sum d = 1..n. \sum e{\in}\{y{\in}\{1..n\}.\ y \leq n$ *div* $d\}.\ (-1)\hat{}(d{+}1)$
$* mangoldt\ e)$
by (*intro sum.cong*) *auto*
also have ... $= (\sum y = 1..n. \sum x \mid x \in \{1..n\} \wedge y \leq n$ *div* $x.\ (-\ 1)\ \hat{}\ (x +$
$1) * mangoldt\ y)$
by (*rule sum.commute-restrict*) *simp-all*
also have ... $= (\sum y = 1..n. \sum x \mid x \in \{1..n\} \wedge x \leq n$ *div* $y.\ (-\ 1)\ \hat{}\ (x +$
$1) * mangoldt\ y)$
by (*intro sum.cong*) (*auto intro*: *div-invert*)
also from $**$ **have** ... $= (\sum y = 1..n. \sum x \in \{1..n$ *div* $y\}.\ (-\ 1)\ \hat{}\ (x + 1) *$
$mangoldt\ y)$
by (*intro sum.cong*) *auto*
also have ... $= (\sum y = 1..n. (\sum x \in \{1..n$ *div* $y\}.\ (-\ 1)\ \hat{}\ (x + 1)) * mangoldt$
$y)$
by (*intro sum.cong*) (*simp-all add*: *sum-distrib-right*)
also have ... $= (\sum y = 1..n.$ (*if odd* $(n$ *div* $y)$ *then 1 else 0*$) * mangoldt\ y)$
by (*intro sum.cong refl*) (*simp-all only*: *sum-minus-one*)
finally show *?thesis* **.**
qed

private lemma *floor-half-interval*:
fixes n d :: *nat*
assumes $d \neq 0$
shows *real* $(n$ *div* $d) -$ *real* $(2 * ((n$ *div* $2)$ *div* $d)) =$ (*if odd* $(n$ *div* $d)$ *then 1*
else 0)
proof −

have $((n\ div\ 2)\ div\ d) = (n\ div\ (2 * d))$
by (*rule div-mult2-eq*[*symmetric*])
also have $\ldots = ((n\ div\ d)\ div\ 2)$
by (*simp add*: *mult-ac div-mult2-eq*)
also have $real\ (n\ div\ d) - real\ (2 * \ldots) = (if\ odd\ (n\ div\ d)\ then\ 1\ else\ 0)$
by (*cases odd* $(n\ div\ d)$, *cases n div d* $= 0$, *simp-all*)
finally show *?thesis* **by** *simp*
qed

lemma *fact-expand-psi*:
$ln\ (fact\ n) - 2 * ln\ (fact\ (n\ div\ 2)) = (\sum d{=}1..n.\ (-1)\hat{}(d+1) * psi\ (n\ div\ d))$
proof −
have $ln\ (fact\ n) - 2 * ln\ (fact\ (n\ div\ 2)) =$
$(\sum d{=}1..n.\ mangoldt\ d * \lfloor n\ /\ d\rfloor) - 2 * (\sum d{=}1..n\ div\ 2.\ mangoldt\ d * \lfloor (n\ div\ 2)\ /\ d\rfloor)$
by (*simp add*: *ln-fact-conv-mangoldt*)
also have $(\sum d{=}1..n\ div\ 2.\ mangoldt\ d * \lfloor real\ (n\ div\ 2)\ /\ d\rfloor) =$
$(\sum d{=}1..n.\ mangoldt\ d * \lfloor real\ (n\ div\ 2)\ /\ d\rfloor)$
by (*rule sum.mono-neutral-left*) (*auto simp*: *floor-unique*[*of 0*])
also have $2 * \ldots = (\sum d{=}1..n.\ mangoldt\ d * 2 * \lfloor real\ (n\ div\ 2)\ /\ d\rfloor)$
by (*simp add*: *sum-distrib-left mult-ac*)
also have $(\sum d{=}1..n.\ mangoldt\ d * \lfloor n\ /\ d\rfloor) - \ldots =$
$(\sum d{=}1..n.\ (mangoldt\ d * \lfloor n\ /\ d\rfloor - mangoldt\ d * 2 * \lfloor real\ (n\ div\ 2)\ /\ d\rfloor))$
by (*simp add*: *sum-subtractf*)
also have $\ldots = (\sum d{=}1..n.\ mangoldt\ d * (\lfloor n\ /\ d\rfloor - 2 * \lfloor real\ (n\ div\ 2)\ /\ d\rfloor))$
by (*simp add*: *algebra-simps*)
also have $\ldots = (\sum d{=}1..n.\ mangoldt\ d * (if\ odd(n\ div\ d)\ then\ 1\ else\ 0))$
by (*intro sum.cong refl*)
(*simp-all add*: *floor-conv-div-nat* [*symmetric*] *floor-half-interval* [*symmetric*])
also have $\ldots = (\sum d{=}1..n.\ (if\ odd(n\ div\ d)\ then\ 1\ else\ 0) * mangoldt\ d)$
by (*simp add*: *mult-ac*)
also from *sum-expand-lemma*[*symmetric*] **have** $\ldots = (\sum d{=}1..n.\ (-1)\hat{}(d+1) * psi\ (n\ div\ d))$ **.**
finally show *?thesis* **.**
qed

end

lemma *psi-expansion-cutoff*:
assumes $m \leq p$
shows $(\sum d{=}1..2{*}m.\ (-1)\hat{}(d+1) * psi\ (n\ div\ d)) \leq (\sum d{=}1..2{*}p.\ (-1)\hat{}(d+1) * psi\ (n\ div\ d))$
$(\sum d{=}1..2{*}p{+}1.\ (-1)\hat{}(d+1) * psi\ (n\ div\ d)) \leq (\sum d{=}1..2{*}m{+}1.\ (-1)\hat{}(d+1) * psi\ (n\ div\ d))$
using *assms*
proof (*induction m rule*: *inc-induct*)
case (*step k*)
have $(\sum d = 1..2 * k.\ (-1)\hat{}(d\ +\ 1) * psi\ (n\ div\ d)) \leq$

```
        (∑ d = 1..2 * Suc k. (−1) ^ (d + 1) * psi (n div d))
    by (simp add: psi-mono div-le-mono2)
  with step.IH(1)
    show (∑ d = 1..2 * k. (−1) ^ (d + 1) * psi (n div d))
      ≤ (∑ d = 1..2 * p. (−1) ^ (d + 1) * psi (n div d))
      by simp
  from step.IH(2)
    have (∑ d = 1..2 * p + 1. (−1) ^ (d + 1) * psi (n div d))
      ≤ (∑ d = 1..2 * Suc k + 1. (−1) ^ (d + 1) * psi (n div d)) .
  also have ... ≤ (∑ d = 1..2 * k + 1. (−1) ^ (d + 1) * psi (n div d))
    by (simp add: psi-mono div-le-mono2)
  finally show (∑ d = 1..2 * p + 1. (−1) ^ (d + 1) * psi (n div d))
      ≤ (∑ d = 1..2 * k + 1. (−1) ^ (d + 1) * psi (n div d)) .
qed simp-all

lemma fact-psi-bound-even:
  assumes even k
  shows   (∑ d=1..k. (−1) ^ (d+1) * psi (n div d)) ≤ ln (fact n) − 2 * ln (fact
(n div 2))
proof −
  have (∑ d=1..k. (−1) ^ (d+1) * psi (n div d)) ≤ (∑ d = 1..n. (− 1) ^ (d +
1) * psi (n div d))
  proof (cases k ≤ n)
    case True
    with psi-expansion-cutoff(1)[of k div 2 n div 2 n]
      have (∑ d=1..2*(k div 2). (−1) ^ (d+1) * psi (n div d))
        ≤ (∑ d = 1..2*(n div 2). (− 1) ^ (d + 1) * psi (n div d))
      by simp
    also from assms have 2*(k div 2) = k
      by simp
    also have (∑ d = 1..2*(n div 2). (− 1) ^ (d + 1) * psi (n div d))
      ≤ (∑ d = 1..n. (− 1) ^ (d + 1) * psi (n div d))
    proof (cases even n)
      case True
      then show ?thesis
        by simp
    next
      case False
      from psi-pos have (∑ d = 1..2*(n div 2). (− 1) ^ (d + 1) * psi (n div d))
          ≤ (∑ d = 1..2*(n div 2) + 1. (− 1) ^ (d + 1) * psi (n div d))
        by simp
      with False show ?thesis
        by simp
    qed
    finally show ?thesis .
  next
    case False
    hence *: n div 2 ≤ (k−1) div 2
      by simp
```

```
    have (∑ d=1..k. (−1) ^(d+1) * psi (n div d)) ≤
          (∑ d=1..2*((k−1) div 2) + 1. (−1) ^(d+1) * psi (n div d))
    proof (cases k = 0)
      case True
      with psi-pos show ?thesis by simp
    next
      case False
      with sum-cl-ivl-Suc[of λd. (−1) ^(d+1) * psi (n div d) 1 k−1]
     have (∑ d=1..k. (−1) ^(d+1) * psi (n div d)) = (∑ d=1..k−1. (−1) ^(d+1)
* psi (n div d))
          + (−1) ^(k+1) * psi (n div k)
        by simp
      also from assms psi-pos have (−1) ^(k+1) * psi (n div k) ≤ 0
        by simp
      also from assms False have k−1 = 2*((k−1) div 2) + 1
        by presburger
      finally show ?thesis by simp
    qed
    also from * psi-expansion-cutoff(2)[of n div 2 (k−1) div 2 n]
     have ... ≤ (∑ d=1..2*(n div 2) + 1. (−1) ^(d+1) * psi (n div d)) by blast
    also have ... ≤ (∑ d = 1..n. (− 1) ^ (d + 1) * psi (n div d))
      by (cases even n) (simp-all add: psi-def)
    finally show ?thesis .
  qed
  also from fact-expand-psi have ... = ln (fact n) − 2 * ln (fact (n div 2)) ..
  finally show ?thesis .
qed

lemma fact-psi-bound-odd:
  assumes odd k
  shows ln (fact n) − 2 * ln (fact (n div 2)) ≤ (∑ d=1..k. (−1) ^(d+1) * psi (n
div d))
proof −
  from fact-expand-psi
    have ln (fact n) − 2 * ln (fact (n div 2)) = (∑ d = 1..n. (− 1) ^ (d + 1) *
psi (n div d)) .
  also have ... ≤ (∑ d=1..k. (−1) ^(d+1) * psi (n div d))
  proof (cases k ≤ n)
    case True
    have (∑ d=1..n. (−1) ^(d+1) * psi (n div d)) ≤ (
           ∑ d=1..2*(n div 2)+1. (−1) ^(d+1) * psi (n div d))
      by (cases even n) (simp-all add: psi-pos)
    also from True assms psi-expansion-cutoff(2)[of k div 2 n div 2 n]
     have ... ≤ (∑ d=1..k. (−1) ^(d+1) * psi (n div d))
       by simp
    finally show ?thesis .
  next
    case False
    have (∑ d=1..n. (−1) ^(d+1) * psi (n div d)) ≤ (∑ d=1..2*((n+1) div 2).
```

```
(−1) ^(d+1) * psi (n div d))
      by (cases even n) (simp-all add: psi-def)
    also from False assms psi-expansion-cutoff(1)[of (n+1) div 2 k div 2 n]
    have (∑ d=1..2*((n+1) div 2). (−1) ^(d+1) * psi (n div d)) ≤ (∑ d=1..2*(k
div 2). (−1) ^(d+1) * psi (n div d))
       by simp
    also from assms have ... ≤ (∑ d=1..k. (−1) ^(d+1) * psi (n div d))
      by (auto elim: oddE simp: psi-pos)
    finally show ?thesis .
  qed
  finally show ?thesis .
qed

lemma fact-psi-bound-2-3:
  psi n − psi (n div 2) ≤ ln (fact n) − 2 * ln (fact (n div 2))
  ln (fact n) − 2 * ln (fact (n div 2)) ≤ psi n − psi (n div 2) + psi (n div 3)
proof −
  show psi n − psi (n div 2) ≤ ln (fact n) − 2 * ln (fact (n div 2))
    by (rule psi-bounds-ln-fact (2))
next
  from fact-psi-bound-odd[of 3 n] have ln (fact n) − 2 * ln (fact (n div 2))
  ≤ (∑ d = 1..3. (− 1) ^ (d + 1) * psi (n div d))
    by simp
  also have ... = psi n − psi (n div 2) + psi (n div 3)
    by (simp add: sum-head-Suc numeral-2-eq-2)
  finally show ln (fact n) − 2 * ln (fact (n div 2)) ≤ psi n − psi (n div 2) + psi
(n div 3) .
qed

lemma ub-ln-1200: ln 1200 ≤ 57 / (8 :: real)
proof −
  have Some (Float 57 (−3)) = ub-ln 8 (Float 1200 0) by code-simp
  from ub-ln(1)[OF this] show ?thesis by simp
qed

lemma psi-double-lemma:
  assumes n ≥ 1200
  shows real n / 6 ≤ psi n − psi (n div 2)
proof −
  from ln-fact-diff-bounds
    have |ln (fact n) − 2 * ln (fact (n div 2)) − real n * ln 2|
        ≤ 4 * ln (real (if n = 0 then 1 else n)) + 3 .
  with assms have ln (fact n) − 2 * ln (fact (n div 2))
        ≥ real n * ln 2 − 4 * ln (real n) − 3
    by simp
  moreover have real n * ln 2 − 4 * ln (real n) − 3 ≥ 2 / 3 * n
  proof (rule overpower-lemma[of λn. 2/3 * n 1200])
    show 2 / 3 * 1200 ≤ 1200 * ln 2 − 4 * ln 1200 − (3::real)
      using ub-ln-1200 ln-2-ge by linarith
```

```
  next
    fix x::real
    assume 1200 ≤ x
    then have 0 < x
      by simp
    show ((λx. x * ln 2 - 4 * ln x - 3 - 2 / 3 * x)
        has-real-derivative ln 2 - 4 / x - 2 / 3) (at x)
      by (rule derivative-eq-intros refl | simp add: ‹0 < x›)+
  next
    fix x::real
    assume 1200 ≤ x
    then have 12 / x ≤ 12 / 1200 by simp
    then have 0 ≤ 0.67 - 4 / x - 2 / 3 by simp
    also have 0.67 ≤ ln (2::real) using ln-2-ge by simp
    finally show 0 ≤ ln 2 - 4 / x - 2 / 3 by simp
  next
    from assms show 1200 ≤ real n
      by simp
  qed
  ultimately have 2 / 3 * real n ≤ ln (fact n) - 2 * ln (fact (n div 2))
    by simp
  with psi-ubound-3-2[of n div 3]
    have n/6 + psi (n div 3) ≤ ln (fact n) - 2 * ln (fact (n div 2))
    by simp
  with fact-psi-bound-2-3[of n] show ?thesis
    by simp
qed

lemma theta-double-lemma:
  assumes n ≥ 1200
  shows theta (n div 2) < theta n
proof -
  from psi-theta[of n div 2] psi-pos[of Discrete.sqrt (n div 2)]
    have theta-le-psi-n-2: theta (n div 2) ≤ psi (n div 2)
    by simp
  have (Discrete.sqrt n * 18) ^2 ≤ 324 * n
    by simp
  from mult-less-cancel2[of 324 n n] assms have 324 * n < n^2
    by (simp add: power2-eq-square)
  with ‹(Discrete.sqrt n * 18) ^2 ≤ 324 * n› have (Discrete.sqrt n*18) ^2 < n^2
    by presburger
  with power2-less-imp-less assms have Discrete.sqrt n * 18 < n
    by blast
  with psi-ubound-3-2[of Discrete.sqrt n] have 2 * psi (Discrete.sqrt n) < n / 6
    by simp
  with psi-theta[of n] have psi-lt-theta-n: psi n - n / 6 < theta n
    by simp
  from psi-double-lemma[OF assms(1)] have psi (n div 2) ≤ psi n - n / 6
    by simp
```

with *theta-le-psi-n-2 psi-lt-theta-n* **show** *?thesis*
by *simp*
qed

0.7 Proof of the main result

lemma *theta-mono*: *mono theta*
by (*auto simp*: *theta-def* [*abs-def*] *intro*!: *monoI sum-mono2*)

lemma *theta-lessE*:
assumes *theta m < theta n m ≥ 1*
obtains *p* **where** *p* ∈ {*m*<..*n*} *prime p*
proof −
from *mono-invE*[*OF theta-mono assms*(*1*)] **have** *m ≤ n* **by** *blast*
hence *theta n = theta m* + (∑ *p*∈{*m*<..*n*}. *if prime p then ln* (*real p*) *else 0*)
unfolding *theta-def* **using** *assms*(*2*)
by (*subst sum.union-disjoint* [*symmetric*]) (*auto simp*: *ivl-disj-un*)
also note *assms*(*1*)
finally have (∑ *p*∈{*m*<..*n*}. *if prime p then ln* (*real p*) *else 0*) ≠ *0* **by** *simp*
from *sum.not-neutral-contains-not-neutral* [*OF this*] **guess** *p* **.**
thus *?thesis* **using** *that*[*of p*] **by** (*auto intro*!: *exI*[*of - p*] *split*: *if-splits*)
qed

theorem *bertrand*:
fixes *n* :: *nat*
assumes *n > 1*
shows ∃ *p*∈{*n*<..<*2*∗*n*}. *prime p*
proof *cases*
assume *n-less*: *n < 600*
define *prime-constants*
where *prime-constants* = {*2*, *3*, *5*, *7*, *13*, *23*, *43*, *83*, *163*, *317*, *631*::*nat*}
from ‹*n > 1*› *n-less* **have** ∃ *p* ∈ *prime-constants*. *n < p* ∧ *p < 2 ∗ n*
unfolding *bex-simps greaterThanLessThan-iff prime-constants-def* **by** *presburger*
moreover have ∀ *p*∈*prime-constants*. *prime p*
unfolding *prime-constants-def ball-simps HOL.simp-thms* **by** (*intro conjI*; *pratt*)
ultimately show *?thesis*
unfolding *greaterThanLessThan-def greaterThan-def lessThan-def* **by** *blast*
next
assume *n*: ¬(*n < 600*)
from *n* **have** *theta n < theta* (*2 ∗ n*) **using** *theta-double-lemma*[*of 2 ∗ n*] **by** *simp*
with *assms* **obtain** *p* **where** *p* ∈ {*n*<..*2*∗*n*} *prime p* **by** (*auto elim*!: *theta-lessE*)
moreover from *assms* **have** ¬*prime* (*2*∗*n*) **by** (*auto dest*!: *prime-product*)
with ‹*prime p*› **have** *p ≠ 2 ∗ n* **by** *auto*
ultimately show *?thesis* **by** *force*
qed

0.8 Proof of Mertens' first theorem

The following proof of Mertens' first theorem was ported from John Harrison's HOL Light proof by Larry Paulson:

lemma *sum-integral-ubound-decreasing'*:
fixes $f :: real \Rightarrow real$
assumes $m \leq n$
and *der*: $\bigwedge x.\ x \in \{of\text{-}nat\ m - 1..of\text{-}nat\ n\} \Longrightarrow (g\ has\text{-}field\text{-}derivative\ f\ x)\ (at\ x)$
and *le*: $\bigwedge x\ y.\ [\![real\ m - 1 \leq x;\ x \leq y;\ y \leq real\ n]\!] \Longrightarrow f\ y \leq f\ x$
shows $(\sum k = m..n.\ f\ (of\text{-}nat\ k)) \leq g\ (of\text{-}nat\ n) - g\ (of\text{-}nat\ m - 1)$
proof −
have $(\sum k = m..n.\ f\ (of\text{-}nat\ k)) \leq (\sum k = m..n.\ g\ (of\text{-}nat(Suc\ k) - 1) - g\ (of\text{-}nat\ k - 1))$
proof (*rule sum-mono*, *clarsimp*)
fix r
assume r: $m \leq r\ r \leq n$
hence $\exists z{>}real\ r - 1.\ z < real\ r \wedge g\ (real\ r) - g\ (real\ r - 1) = (real\ r - (real\ r - 1)) * f\ z$
using *assms* **by** (*intro MVT2*) *auto*
hence $\exists z{\in}\{of\text{-}nat\ r - 1..of\text{-}nat\ r\}.\ g\ (real\ r) - g\ (real\ r - 1) = f\ z$ **by** *auto*
then obtain *u*::*real* **where** *u*: $u \in \{of\text{-}nat\ r - 1..of\text{-}nat\ r\}$
and *eq*: $g\ r - g\ (of\text{-}nat\ r - 1) = f\ u$ **by** *blast*
have $real\ m \leq u + 1$
using *r u* **by** *auto*
then have $f\ (of\text{-}nat\ r) \leq f\ u$
using *r*(*2*) **and** *u* **by** (*intro le*) *auto*
then show $f\ (of\text{-}nat\ r) \leq g\ r - g\ (of\text{-}nat\ r - 1)$
by (*simp add*: *eq*)
qed
also have $\ldots \leq g\ (of\text{-}nat\ n) - g\ (of\text{-}nat\ m - 1)$
using ‹$m \leq n$› **by** (*subst sum-Suc-diff*) *auto*
finally show *?thesis* **.**
qed

lemma *Mertens-lemma*:
assumes $n \neq 0$
shows $|(\sum d = 1..n.\ mangoldt\ d\ /\ real\ d) - ln\ n| \leq 4$
proof −
have ∗: $[\![abs(s' - nl + n) \leq a;\ abs(s' - s) \leq (k - 1) * n - a]\!] \Longrightarrow abs(s - nl) \leq n * k$ **for** *s' s k nl a*::*real*
by (*auto simp*: *algebra-simps abs-if split*: *if-split-asm*)
have *le*: $|(\sum d{=}1..n.\ mangoldt\ d * floor\ (n\ /\ d)) - n * ln\ n + n| \leq 1 + ln\ n$
using *ln-fact-bounds ln-fact-conv-mangoldt assms* **by** *simp*
have $|real\ n * ((\sum d = 1..n.\ mangoldt\ d\ /\ real\ d) - ln\ n)| = |((\sum d = 1..n.\ real\ n * mangoldt\ d\ /\ real\ d) - n * ln\ n)|$
by (*simp add*: *algebra-simps sum-distrib-left*)
also have $\ldots \leq real\ n * 4$
proof (*rule* ∗ [*OF le*])

```
    have |(∑ d = 1..n. mangoldt d * ⌊n / d⌋) - (∑ d = 1..n. n * mangoldt d /
d)|
        = |∑ d = 1..n. mangoldt d * (⌊n / d⌋ - n / d)|
      by (simp add: sum-subtractf algebra-simps)
    also have ... ≤ psi n (is |?sm| ≤ ?rhs)
    proof -
      have -?sm = (∑ d = 1..n. mangoldt d * (n/d - ⌊n/d⌋))
        by (simp add: sum-subtractf algebra-simps)
      also have ... ≤ (∑ d = 1..n. mangoldt d * 1)
        by (intro sum-mono mult-left-mono mangoldt-pos) linarith
      finally have -?sm ≤ ?rhs by (simp add: psi-def)
      moreover
      have ?sm ≤ 0
        using mangoldt-pos by (simp add: mult-le-0-iff sum-nonpos)
      ultimately show ?thesis
        by (simp add: abs-if)
    qed
    also have ... ≤ 3/2 * real n
      by (rule psi-ubound-3-2)
    also have ...≤ (4 - 1) * real n - (1 + ln n)
      using ln-le-minus-one [of n] assms by (simp add: divide-simps)
    finally
    show |(∑ d = 1..n. mangoldt d * real-of-int ⌊real n / real d⌋) -
          (∑ d = 1..n. real n * mangoldt d / real d)|
         ≤ (4 - 1) * real n - (1 + ln n) .
  qed
  finally have |real n * ((∑ d = 1..n. mangoldt d / real d) - ln n)| ≤ real n * 4
.
  then show ?thesis
    using assms mult-le-cancel-left-pos by (simp add: abs-mult)
qed

lemma Mertens-mangoldt-versus-ln:
  assumes I ⊆ {1..n}
  shows |(∑ i∈I. mangoldt i / i) - (∑ p | prime p ∧ p ∈ I. ln p / p)| ≤ 3
        (is |?lhs| ≤ 3)
proof (cases n = 0)
  case True
  with assms show ?thesis by simp
next
  case False
    have finite I
      using assms finite-subset by blast
    have 0 ≤ (∑ i∈I. mangoldt i / i - (if prime i then ln i / i else 0))
      using mangoldt-pos by (simp add: sum-nonneg)
    moreover have ... ≤ (∑ i = 1..n. mangoldt i / i - (if prime i then ln i / i
else 0))
      using assms by (intro sum-mono2) (auto simp: mangoldt-pos)
    ultimately have *: |∑ i∈I. mangoldt i / i - (if prime i then ln i / i else 0)|
```

$\leq |\sum i = 1..n.\ mangoldt\ i\ /\ i - (if\ prime\ i\ then\ ln\ i\ /\ i\ else\ 0)|$
by *linarith*
moreover have *?lhs* = $(\sum i \in I.\ mangoldt\ i\ /\ i - (if\ prime\ i\ then\ ln\ i\ /\ i\ else\ 0))$
$(\sum i = 1..n.\ mangoldt\ i\ /\ i - (if\ prime\ i\ then\ ln\ i\ /\ i\ else\ 0))$
$= (\sum d = 1..n.\ mangoldt\ d\ /\ d) - (\sum p \mid prime\ p \wedge p \in \{1..n\}.\ ln\ p\ /\ p)$
using *sum.inter-restrict* [*of* - $\lambda i.\ ln\ (real\ i)\ /\ i$ *Collect prime*, *symmetric*]
by (*force simp*: *sum-subtractf* ‹*finite I*› *intro*: *sum.cong*)+
ultimately have $|?lhs| \leq |(\sum d = 1..n.\ mangoldt\ d\ /\ d) - (\sum p \mid prime\ p \wedge p \in \{1..n\}.\ ln\ p\ /\ p)|$ **by** *linarith*
also have ... ≤ 3
proof −
have *eq-sm*: $(\sum i = 1..n.\ mangoldt\ i\ /\ i) = (\sum i \in \{p\hat{}k \mid p\ k.\ prime\ p \wedge p\hat{}k \leq n \wedge k \geq 1\}.\ mangoldt\ i\ /\ i)$
by (*intro sum.mono-neutral-right*)
(*auto simp*: *mangoldt-def primepow-def dest*: *prime-ge-1-nat split*: *if-split-asm*)
have $(\sum i = 1..n.\ mangoldt\ i\ /\ i) - (\sum p \mid prime\ p \wedge p \in \{1..n\}.\ ln\ p\ /\ p) =$
$(\sum i \in \{p\hat{}k \mid p\ k.\ prime\ p \wedge p\hat{}k \leq n \wedge k \geq 2\}.\ mangoldt\ i\ /\ i)$
proof −
have *eq*: $\{p\ \hat{}\ k \mid p\ k.\ prime\ p \wedge p\ \hat{}\ k \leq n \wedge 1 \leq k\} = \{p\ \hat{}\ k \mid p\ k.\ prime\ p \wedge p\ \hat{}\ k \leq n \wedge 2 \leq k\} \cup \{p.\ prime\ p \wedge p \in \{1..n\}\}$
by (*force simp*: *prime-ge-Suc-0-nat prime-power-iff*)
have *eqln*: $(\sum p \mid prime\ p \wedge p \in \{1..n\}.\ ln\ p\ /\ p) = (\sum p \mid prime\ p \wedge p \in \{1..n\}.\ mangoldt\ p\ /\ p)$
by (*rule sum.cong*) *auto*
have $(\sum i \in \{p\hat{}k \mid p\ k.\ prime\ p \wedge p\hat{}k \leq n \wedge k \geq 1\}.\ mangoldt\ i\ /\ i) = (\sum i \in \{p\ \hat{}\ k \mid p\ k.\ prime\ p \wedge p\ \hat{}\ k \leq n \wedge 2 \leq k\} \cup \{p.\ prime\ p \wedge p \in \{1..n\}\}.\ mangoldt\ i\ /\ i)$ **by** (*subst eq*) *simp-all*
also have ... $= (\sum i \in \{p\hat{}k \mid p\ k.\ prime\ p \wedge p\hat{}k \leq n \wedge k \geq 2\}.\ mangoldt\ i\ /\ i) + (\sum p \mid prime\ p \wedge p \in \{1..n\}.\ mangoldt\ p\ /\ p)$
by (*intro sum.union-disjoint*) (*auto simp*: *prime-power-iff finite-nat-set-iff-bounded-le*)
also have ... $= (\sum i \in \{p\hat{}k \mid p\ k.\ prime\ p \wedge p\hat{}k \leq n \wedge k \geq 2\}.\ mangoldt\ i\ /\ i) + (\sum p \mid prime\ p \wedge p \in \{1..n\}.\ ln\ p\ /\ p)$ **by** (*simp only*: *eqln*)
finally show *?thesis*
using *eq-sm* **by** *auto*
qed
have $(\sum p \mid prime\ p \wedge p \in \{1..n\}.\ ln\ p\ /\ p) \leq (\sum p \mid prime\ p \wedge p \in \{1..n\}.\ mangoldt\ p\ /\ p)$
using *mangoldt-pos* **by** (*auto intro*: *sum-mono*)
also have ... $\leq (\sum i = Suc\ 0..n.\ mangoldt\ i\ /\ i)$
by (*intro sum-mono2*) (*auto simp*: *mangoldt-pos*)
finally have $0 \leq (\sum i = 1..n.\ mangoldt\ i\ /\ i) - (\sum p \mid prime\ p \wedge p \in \{1..n\}.\ ln\ p\ /\ p)$

```
    by simp
  moreover have (∑ i = 1..n. mangoldt i / i) − (∑ p | prime p ∧ p ∈ {1..n}.
ln p / p) ≤ 3
                (is ?M − ?L ≤ 3)
  proof −
    have ∗: ∃ q. ∃ j∈{1..n}. prime q ∧ 1 ≤ q ∧ q ≤ n ∧
          (q ^ j = p ^ k ∧ mangoldt (p ^ k) / real p ^ k ≤ ln (real q) / real q
^ j)
      if prime p p ^ k ≤ n 1 ≤ k for p k
    proof −
      have mangoldt (p ^ k) / real p ^ k ≤ ln p / p ^ k
        using that by (simp add: divide-simps)
      moreover have p ≤ n
        using that self-le-power[of p k] by (simp add: prime-ge-Suc-0-nat)
      moreover have k ≤ n
      proof −
        have k < 2^k
        using of-nat-less-two-power real-of-nat-less-numeral-power-cancel-iff by
blast
        also have ... ≤ p^k
          by (simp add: power-mono prime-ge-2-nat that)
        also have ... ≤ n
          by (simp add: that)
        finally show ?thesis by (simp add: that)
      qed
      ultimately show ?thesis
        using prime-ge-1-nat that by force
    qed
    have finite: finite {p ^ k |p k. prime p ∧ p ^ k ≤ n ∧ 1 ≤ k}
      by (rule finite-subset[of - {..n}]) auto
    have ?M ≤ (∑ (x, k)∈{p. prime p ∧ p ∈ {1..n}} × {1..n}. ln (real x) /
real x ^ k)
      by (subst eq-sm, intro sum-le-included [where i = λ(p,k). p^k])
        (insert ∗ finite, auto)
    also have ... = (∑ p | prime p ∧ p ∈ {1..n}. (∑ k = 1..n. ln p / p^k))
      by (subst sum.Sigma) auto
    also have ... = ?L + (∑ p | prime p ∧ p ∈ {1..n}. (∑ k = 2..n. ln p /
p^k))
      by (simp add: comm-monoid-add-class.sum.distrib sum-head-Suc numeral-2-eq-2)
    finally have ?M − ?L ≤ (∑ p | prime p ∧ p ∈ {1..n}. (∑ k = 2..n. ln p
/ p^k))
      by (simp add: algebra-simps)
    also have ... = (∑ p | prime p ∧ p ∈ {1..n}. ln p ∗ (∑ k = 2..n. inverse
p ^ k))
      by (simp add: field-simps sum-distrib-left)
    also have ... = (∑ p | prime p ∧ p ∈ {1..n}.
                  ln p ∗ (((inverse p)^2 − inverse p ^ Suc n) / (1 − inverse p)))
      by (intro sum.cong refl) (simp add: sum-gp)
    also have ... ≤ (∑ p | prime p ∧ p ∈ {1..n}. ln p ∗ inverse (real (p ∗ (p
```

```
− 1))))
      by (intro sum-mono mult-left-mono)
        (auto simp: divide-simps power2-eq-square of-nat-diff mult-less-0-iff)
    also have ... ≤ (∑ p = 2..n. ln p * inverse (real (p * (p − 1))))
      by (rule sum-mono2) (use prime-ge-2-nat in auto)
    also have ... ≤ (∑ i = 2..n. ln i / (i − 1)^2)
      unfolding divide-inverse power2-eq-square mult.assoc
      by (auto intro: sum-mono mult-left-mono mult-right-mono)
    also have ... ≤ 3
    proof (cases n ≥ 3)
      case False then show ?thesis
      proof (cases n ≥ 2)
        case False then show ?thesis by simp
      next
        case True
        then have n = 2 using False by linarith
        with ln-le-minus-one [of 2] show ?thesis by simp
      qed
    next
      case True
      have (∑ i = 3..n. ln (real i) / (real (i − Suc 0))^2)
            ≤ (ln (of-nat n − 1)) − (ln (of-nat n)) − (ln (of-nat n) / (of-nat n
− 1)) + 2 * ln 2
      proof −
        have 1: ((λz. ln (z − 1) − ln z − ln z / (z − 1)) has-field-derivative
ln x / (x − 1)^2) (at x)
          if x: x ∈ {2..real n} for x
          by (rule derivative-eq-intros | rule refl |
              (use x in ‹force simp: power2-eq-square divide-simps›))+
        have 2: ln y / (y − 1)^2 ≤ ln x / (x − 1)^2 if xy: 2 ≤ x x ≤ y y ≤ real
n for x y
        proof (cases x = y)
          case False
          define f' :: real ⇒ real
            where f' = (λu. ((u − 1)^2 / u − ln u * (2 * u − 2)) / (u − 1) ^ 4)
          have f'-altdef: f' u = inverse u * inverse ((u − 1)^2) − 2 * ln u / (u
− 1) ^ 3
            if u: u ∈ {x..y} for u::real unfolding f'-def using u
          by (simp add: eval-nat-numeral divide-simps) (simp add: algebra-simps)?
          have deriv: ((λz. ln z / (z − 1)^2) has-field-derivative f' u) (at u)
            if u: u ∈ {x..y} for u::real unfolding f'-def
                by (rule derivative-eq-intros refl | (use u xy in ‹force simp:
divide-simps›))+
          hence ∃z>x. z < y ∧ ln y / (y − 1)^2 − ln x / (x − 1)^2 = (y − x) *
f' z
            using xy and ‹x ≠ y› by (intro MVT2) auto
          then obtain ξ::real where x < ξ ξ < y
            and ξ: ln y / (y − 1)^2 − ln x / (x − 1)^2 = (y − x) * f' ξ by blast
          have f' ξ ≤ 0
```

```
        proof −
          have 2/3 ≤ ln (2::real) by (fact ln-2-ge')
          also have ... ≤ ln ξ
            using ‹x < ξ› xy by auto
          finally have 1 ≤ 2 * ln ξ by simp
          then have *: ξ ≤ ξ * (2 * ln ξ)
            using ‹x < ξ› xy by auto
          hence ξ − 1 ≤ ln ξ * 2 * ξ by (simp add: algebra-simps)
          hence 1 / (ξ * (ξ − 1)^2) ≤ ln ξ * 2 / (ξ − 1) ^ 3
            using xy ‹x < ξ› by (simp add: divide-simps power-eq-if)
        thus ?thesis using xy ‹x < ξ› ‹ξ < y› by (subst f'-altdef) (auto simp:
divide-simps)
      qed
      then have (ln y / (y − 1)^2 − ln x / (x − 1)^2) ≤ 0
        using ‹x ≤ y› by (simp add: mult-le-0-iff ξ)
      then show ?thesis by simp
    qed simp-all
    show ?thesis
      using sum-integral-ubound-decreasing'
          [OF ‹3 ≤ n›, of λz. ln(z−1) − ln z − ln z / (z − 1) λz. ln z /
(z−1)^2]
        1 2 ‹3 ≤ n›
      by (auto simp: in-Reals-norm of-nat-diff)
  qed
  also have ... ≤ 2
  proof −
    have ln (real n − 1) − ln n ≤ 0  0 ≤ ln n / (real n − 1)
      using ‹3 ≤ n› by auto
    then have ln (real n − 1) − ln n − ln n / (real n − 1) ≤ 0
      by linarith
    with ln-2-less-1 show ?thesis by linarith
  qed
  also have ... ≤ 3 − ln 2
    using ln-2-less-1 by (simp add: algebra-simps)
  finally show ?thesis
    using True by (simp add: algebra-simps sum-head-Suc [of 2 n])
qed
  finally show ?thesis .
qed
  ultimately show ?thesis
    by linarith
qed
finally show ?thesis .
qed

proposition Mertens:
  assumes n ≠ 0
  shows |(∑ p | prime p ∧ p ≤ n. ln p / of-nat p) − ln n| ≤ 7
proof −
```

have $|(\sum d = 1..n.\ mangoldt\ d\ /\ real\ d) - (\sum p \mid prime\ p \wedge p \in \{1..n\}.\ ln\ (real\ p)\ /\ real\ p)|$
$\leq 7 - 4$ **using** *Mertens-mangoldt-versus-ln* [*of* $\{1..n\}$ n] **by** *simp-all*
also have $\{p.\ prime\ p \wedge p \in \{1..n\}\} = \{p.\ prime\ p \wedge p \leq n\}$
using *atLeastAtMost-iff prime-ge-1-nat* **by** *blast*
finally have $|(\sum d = 1..n.\ mangoldt\ d\ /\ real\ d) - (\sum p{\in}\ldots.\ ln\ (real\ p)\ /\ real\ p)| \leq 7 - 4$ **.**
moreover from *assms* **have** $|(\sum d = 1..n.\ mangoldt\ d\ /\ real\ d) - ln\ n| \leq 4$
by (*rule Mertens-lemma*)
ultimately show *?thesis* **by** *linarith*
qed

end

References

[1] J. Harrison. HOL Light, Bertrand's postulate.
https://github.com/jrh13/hol-light/blob/master/100/bertrand.ml.