

Concrete bounds for Chebyshev's prime counting functions

Manuel Eberl

September 16, 2024

Abstract

This entry derives explicit lower and upper bounds for Chebyshev's prime counting functions

$$\psi(x) = \sum_{\substack{p^k \leq x \\ k > 0}} \log p \qquad \vartheta(x) = \sum_{p \leq x} \log p .$$

Concretely, the following inequalities are proven:

- $\psi(x) \geq 0.9x$ for $x \geq 41$
- $\vartheta(x) \geq 0.82x$ if $x \geq 97$
- $\vartheta(x) \leq \psi(x) \leq 1.2x$ if $x \geq 0$

The proofs work by careful estimation of $\psi(x)$, with Stirling's formula as a starting point, to prove the bound for all $x \geq x_0$ with a concrete x_0 , followed by brute-force approximation for all x below x_0 .

An easy corollary of this is *Bertrand's postulate*, i.e. the fact that for any $x > 1$ the interval $(x, 2x)$ contains at least one prime (a fact that has already been shown in the AFP using weaker bounds for ψ and ϑ).

Contents

| | | |
|----------|---|----------|
| 1 | Concrete bounds for Chebyshev's prime counting functions | 2 |
| 1.1 | Brute-force checking of bounds for ψ and ϑ | 2 |
| 1.1.1 | Computing powers of a number | 2 |
| 1.1.2 | Computing prime powers | 4 |
| 1.1.3 | A generic checking function | 6 |
| 1.1.4 | The ϑ function | 11 |
| 1.1.5 | The ψ function | 15 |
| 1.2 | Auxiliary material | 19 |
| 1.3 | Bounds for the remainder in Stirling's approximation | 20 |
| 1.4 | Approximating ψ | 23 |
| 1.5 | Final results | 34 |

1 Concrete bounds for Chebyshev's prime counting functions

```
theory Chebyshev_Prime_Exhaust
imports
  "HOL-Decision_Procs.Approximation"
  "HOL-Library.Code_Target_Natural"
  "Prime_Number_Theorem.Prime_Counting_Functions"
begin
```

The well-known Prime Number Theorem states that $\psi(x) \sim \theta(x) \sim x$, i.e. that both $\psi(x)$ and $\vartheta(x)$ are bounded by $(1 \pm \varepsilon)x$ for sufficiently large x for any $\varepsilon > 0$. However, these are asymptotic bounds without giving any concrete information on how ψ and ϑ behave for small x , or even how big x must be until these bound shold.

To complement this, we shall prove some concrete, non-asymptotic bounds. Concretely:

- $\psi(x) \geq 0.9x$ if $x \geq 41$
- $\theta(x) \geq 0.82x$ if $x \geq 97$
- $\theta(x) \leq \psi(x) \leq 1.2x$ if $x \geq 0$

Our formalisation loosely follows a blog entry by A.W. Walker: <https://awwalker.com/2017/02/05/notes-on-the-chebyshev-theorem/>

1.1 Brute-force checking of bounds for ψ and ϑ

1.1.1 Computing powers of a number

```
function powers_below_aux :: "nat  $\Rightarrow$  nat  $\Rightarrow$  nat  $\Rightarrow$  nat list" where
  "powers_below_aux ub n acc = (if acc = 0  $\vee$  n  $\leq$  1  $\vee$  acc > ub then []
else
  acc # powers_below_aux ub n (acc * n))"
  by auto
termination
  by (relation "Wellfounded.measure ( $\lambda$ (ub, n, acc). 1 + ub - acc)")
    (auto intro!: diff_less_mono2)

lemmas [simp del] = powers_below_aux.simps

lemma set_powers_below_aux:
  assumes "acc > 0" "n > 1"
  shows "set (powers_below_aux ub n acc) = range ( $\lambda$ i. acc * n  $\wedge$  i)
 $\cap$  {..ub}"
  using assms
proof (induction ub n acc rule: powers_below_aux.induct)
```

```

case (1 ub n acc)
show ?case
proof (cases "acc > ub")
  case True
  have "range ( $\lambda i. \text{acc} * n^i$ )  $\cap \{..ub\} = \{\}$ "
  proof (intro equalityI subsetI)
    fix k assume "k  $\in$  range ( $\lambda i. \text{acc} * n^i$ )  $\cap \{..ub\}$ "
    then obtain i where "acc *  $n^i \leq ub$ "
      by auto
    also have "ub < acc *  $n^0$ "
      using True by simp
    finally have " $n^i < n^0$ "
      using <acc > 0> by (subst (asm) mult_less_cancel1) auto
    hence "i < 0"
      by (subst (asm) power_strict_increasing_iff) (use <n > 1> in auto)
    thus "k  $\in \{\}$ "
      by simp
  qed auto
  thus ?thesis
    using True by (auto simp: powers_below_aux.simps)
next
  case False
  have "set (powers_below_aux ub n acc) = insert acc (set (powers_below_aux
ub n (acc * n)))"
    using False "1.prem" by (subst powers_below_aux.simps) auto
  also have "set (powers_below_aux ub n (acc * n)) = range ( $\lambda i. \text{acc} * n^{Suc\ i}$ )  $\cap \{..ub\}$ "
    by (subst "1.IH") (use "1.prem" False in <auto simp: mult_ac>)
  also have "insert acc (range ( $\lambda i. \text{acc} * n^{Suc\ i}$ )  $\cap \{..ub\}) =$ 
    range ( $\lambda i. \text{acc} * n^i$ )  $\cap \{..ub\}$ " (is "insert acc ?lhs
= ?rhs")
  proof (intro equalityI subsetI)
    fix x assume "x  $\in$  insert acc ?lhs"
    thus "x  $\in$  ?rhs" using False
      by (auto intro: rev_image_eqI[of 0] rev_image_eqI[of "Suc i" for
i])
  next
    fix x assume "x  $\in$  ?rhs"
    then obtain i where i: "x = acc *  $n^i$ " and le: "acc *  $n^i \leq ub$ "
      by auto
    show "x  $\in$  insert acc ?lhs"
    proof (cases "i = 0")
      case False
      hence "x  $\in$  ?lhs"
        by (intro IntI rev_image_eqI[of "i-1"]) (use i le in auto)
      thus ?thesis
        by blast
    qed (use i le in auto)

```

```

    qed
  finally show ?thesis .
  qed
qed

```

```

definition powers_below :: "nat  $\Rightarrow$  nat  $\Rightarrow$  nat list" where
  "powers_below ub n = powers_below_aux ub n n"

```

```

lemma set_powers_below:
  assumes "n > 1"
  shows "set (powers_below ub n) = ( $\lambda$ i. n  $\wedge$  i) ' {1..}  $\cap$  {..ub}"
proof -
  have "set (powers_below ub n) = range ( $\lambda$ i. n * n  $\wedge$  i)  $\cap$  {..ub}"
    unfolding powers_below_def
    by (rule set_powers_below_aux) (use assms in auto)
  also have "range ( $\lambda$ i. n * n  $\wedge$  i) = ( $\lambda$ i. n  $\wedge$  i) ' Suc ' UNIV"
    by (simp add: image_image o_def)
  also have "bij_betw Suc UNIV {1..}"
    by (rule bij_betwI[of _ _ _ " $\lambda$ i. i - 1"]) auto
  hence "Suc ' UNIV = {1..}"
    by (simp add: bij_betw_def)
  finally show ?thesis .
qed

```

```

lemma distinct_powers_below_aux:
  assumes "n > 1" "acc > 0"
  shows "distinct (powers_below_aux ub n acc)"
  using assms
  by (induction ub n acc rule: powers_below_aux.induct; subst powers_below_aux.simps)
    (auto simp: set_powers_below_aux)

```

```

lemma distinct_powers_below: "n > 1  $\implies$  distinct (powers_below ub n)"
  unfolding powers_below_def by (rule distinct_powers_below_aux) auto

```

```

lemma hd_powers_below_aux:
  assumes "acc  $\leq$  ub" "n > 1" "acc > 0"
  shows "hd (powers_below_aux ub n acc) = acc"
  by (subst powers_below_aux.simps) (use assms in auto)

```

```

lemma hd_powers_below:
  assumes "n  $\leq$  ub" "n > 1"
  shows "hd (powers_below ub n) = n"
  unfolding powers_below_def by (subst hd_powers_below_aux) (use assms
in auto)

```

1.1.2 Computing prime powers

```

definition prime_powers_upto :: "nat  $\Rightarrow$  (nat  $\times$  nat) list" where
  "prime_powers_upto n =

```

```

      sort_key fst (concat (map (λp. map (λk. (k, p)) (powers_below n p))
        (primes_upto n))))"

lemma map_key_sort_key: "map f (sort_key f xs) = sort (map f xs)"
proof -
  have [simp]: "map f (insort_key f x xs) = insort (f x) (map f xs)" for
x xs
    by (induction xs) auto
  have [simp]: "map f (foldr (insort_key f) xs acc) =
    foldr insort (map f xs) (map f acc)" for acc
    by (induction xs arbitrary: acc) auto
  show ?thesis
    unfolding sort_key_def by simp
qed

lemma distinct_prime_powers_upto:
  "distinct (map fst (prime_powers_upto n))"
proof -
  have inj: "inj_on (powers_below n) {p. prime p ∧ p ≤ n}"
  proof
    fix p q assume pq: "p ∈ {p. prime p ∧ p ≤ n}" "q ∈ {p. prime p
  ∧ p ≤ n}"
    assume eq: "powers_below n p = powers_below n q"
    from eq have "hd (powers_below n p) = hd (powers_below n q)"
      by simp
    thus "p = q"
      using pq by (simp add: hd_powers_below prime_gt_Suc_0_nat)
  qed

  have "distinct (concat (map (powers_below n) (primes_upto n)))"
  proof (rule distinct_concat, goal_cases)
    case 1
    thus ?case
      unfolding distinct_map using inj
      by (simp add: set_primes_upto conj_commute)
  next
    case (2 ys)
    thus ?case
      by (auto simp: distinct_powers_below set_primes_upto prime_gt_Suc_0_nat)
  next
    case (3 ys zs)
    thus ?case
      by (auto simp: set_primes_upto set_powers_below prime_gt_Suc_0_nat
prime_power_inj'')
  qed
  thus ?thesis
    by (simp add: prime_powers_upto_def map_key_sort_key map_concat o_def)
qed

```

```

lemma sorted_prime_powers_upto:
  "sorted (map fst (prime_powers_upto n))"
  by (simp add: prime_powers_upto_def)

lemma set_prime_powers_upto:
  "set (prime_powers_upto n) = {(q, aprimedivisor q) | q. primepow q ∧
q ≤ n}"
proof -
  have "set (prime_powers_upto n) =
    (⋃ p ∈ {p. p ≤ n ∧ prime p}. (λx. (x, p)) ' ((λi. p ^ i) ' {1..}
    ∩ {..n}))"
    by (simp add: prime_powers_upto_def set_primes_upto set_powers_below
prime_gt_Suc_0_nat)
  also have "... = {(q, aprimedivisor q) | q. primepow q ∧ q ≤ n}"
    (is "?lhs = ?rhs")
  proof (intro equalityI subsetI)
    fix qp assume qp: "qp ∈ ?lhs"
    then obtain q p where [simp]: "qp = (q, p)"
      by (cases qp)
    from qp obtain i where i: "prime p" "p ≤ n" "p ^ i ≤ n" "q = p
    ^ i" "i ≥ 1"
      by auto
    show "qp ∈ ?rhs"
      using i by (auto simp: aprimedivisor_prime_power)
  next
    fix qp assume qp: "qp ∈ ?rhs"
    then obtain q p where [simp]: "qp = (q, p)"
      by (cases qp)
    from qp have "primepow q"
      by auto
    then obtain p' i where i: "prime p'" "q = p' ^ i" "i > 0"
      by (auto simp: primepow_def)
    have [simp]: "p' = p"
      using qp i by (auto simp: aprimedivisor_prime_power)
    have "p ^ 1 ≤ p ^ i"
      by (rule power_increasing) (use i prime_gt_0_nat[of p] in auto)
    also have "... ≤ n"
      using i qp by simp
    finally have "p ≤ n"
      by simp
    with i qp show "qp ∈ ?lhs"
      by auto
  qed
  finally show ?thesis .
qed

```

1.1.3 A generic checking function

locale chebyshev_check =

```

fixes f :: "nat  $\Rightarrow$  real"
  and F :: "nat  $\Rightarrow$  'a  $\Rightarrow$  float"
  and A :: "nat set"
  and plus :: "nat  $\Rightarrow$  float  $\Rightarrow$  float  $\Rightarrow$  float"
  and rel :: "real  $\Rightarrow$  real  $\Rightarrow$  bool"
  and P :: "nat  $\Rightarrow$  real  $\Rightarrow$  bool"
  and num :: "'a  $\Rightarrow$  nat"
assumes plus: " $\bigwedge$ prec. rel X x  $\implies$  rel Y y  $\implies$  rel (plus prec X Y)
(x + y)"
assumes P_rel: " $\bigwedge$ x y k. P k x  $\implies$  rel x y  $\implies$  P k y"
assumes rel_0: "rel 0 0"
assumes A: "0  $\notin$  A"
begin

definition S where "S n = ( $\sum$  k  $\in$  A  $\cap$  {..definition S' where "S' n = ( $\sum$  k  $\in$  A  $\cap$  {..context
  fixes prec :: nat
begin

function check_aux :: "'a list  $\Rightarrow$  nat  $\Rightarrow$  nat  $\Rightarrow$  float  $\Rightarrow$  nat  $\Rightarrow$  bool"
where
  "check_aux ps lb ub acc n = (if n > ub then True else
    (let (acc', ps') =
      (if ps  $\neq$  []  $\wedge$  num (hd ps) = n then
        (plus prec acc (F prec (hd ps)), tl ps)
      else (acc, ps))
      in (n < lb  $\vee$  P n (real_of_float acc'))  $\wedge$  check_aux ps' lb ub acc'
(n+1)))"
  by auto
termination
  by (relation "Wellfounded.measure ( $\lambda$ (_, _, ub, _, n). Suc ub - n)"
    (auto split: if_splits))

definition check :: "'a list  $\Rightarrow$  nat  $\Rightarrow$  nat  $\Rightarrow$  bool" where
  "check xs lb ub =
    check_aux xs lb ub 0 (if xs = [] then lb else min lb (num (hd xs)))"

lemmas [simp del] = check_aux.simps

lemma check_aux_correct:
  assumes "sorted (map num ps)" "distinct (map num ps)"
  assumes " $\bigwedge$ p. p  $\leq$  ub  $\implies$  p  $\in$  num ' set ps  $\longleftrightarrow$  p  $\in$  A  $\wedge$  p  $\geq$  n"
  assumes " $\bigwedge$ x. x  $\in$  set ps  $\implies$  rel (real_of_float (F prec x)) (f (num
x))"
  assumes "rel (real_of_float acc) (S' n)"
  assumes "check_aux ps lb ub acc n"
  assumes "k  $\in$  {max lb n..ub}"

```

```

shows "P k (S k)"
using assms
proof (induction ps lb ub acc n rule: check_aux.induct)
  case (1 ps lb ub acc n)
  hence "n ≤ ub"
  by auto
  define ps' where "ps' = (if ps = [] ∨ num (hd ps) ≠ n then ps else
tl ps)"
  define acc' where "acc' = (if ps = [] ∨ num (hd ps) ≠ n then acc else
plus prec acc (F prec (hd ps)))"

  have acc': "rel (real_of_float acc') (S n)"
  proof (cases "n ∈ A")
    case False
    hence "acc' = acc" using "1.prem1"(3)[of n] <n ≤ ub>
      by (cases ps) (auto simp: acc'_def)
    hence "rel (real_of_float acc') (S' n)"
      using "1.prem1"(5) by simp
    also from False have "A ∩ {...<n} = A ∩ {...n}"
      using nless_le by blast
    hence "S' n = S n"
      by (simp add: S_def S'_def)
    finally show ?thesis .
  next
  case True
  hence "n ∈ num ' set ps" "n > 0"
    using "1.prem1"(3)[of n] <n ≤ ub> A by (auto intro: Nat.gr0I)
  have *: "num p ≥ n" if "p ∈ set ps" for p
    using "1.prem1"(3)[of "num p"] that <n ≤ ub>
    by (cases "num p ≤ ub") auto
  from <n ∈ num ' set ps> obtain x
    where ps_eq: "ps = x # ps'" "num x = n"
    using <sorted (map num ps)> <distinct (map num ps)> *
    by (cases ps) (fastforce simp: ps'_def)+
  have "acc' = plus prec acc (F prec x)"
    using ps_eq by (auto simp: acc'_def)
  also have "rel (real_of_float ...) (S' n + f (num x))"
    by (intro plus "1.prem1" <n > 0>) (auto simp: ps_eq)
  also have "... = sum f (insert n (A ∩ {...<n}))"
    unfolding S'_def by (subst sum.insert) (auto simp: ps_eq)
  also have "insert n (A ∩ {...<n}) = A ∩ {...n}"
    using True by auto
  also have "sum f ... = S n"
    by (simp add: S_def)
  finally show ?thesis .
qed

show ?case
proof (cases "n = k")

```



```

case True
have "P k (real_of_float acc')"
  using "1.prem" (6,7)
  by (subst (asm) check_aux.simps) (use True in <auto simp: acc'_def>)
moreover have "rel (real_of_float acc') (S n)"
  by fact
ultimately show ?thesis
  using True P_rel by simp
next
case False
show ?thesis
proof (rule "1.IH"[of "(acc', ps')", OF _ _ refl])
  show "sorted (map num ps')"
    using <sorted (map num ps)>
    by (auto simp: ps'_def sorted_tl map_tl)
  show "distinct (map num ps')"
    using <distinct (map num ps)>
    by (auto simp: ps'_def distinct_tl map_tl)
  show "(p ∈ num ' set ps') = (p ∈ A ∧ n + 1 ≤ p)" if p: "p ≤ ub"
for p
  proof (cases "n ∈ A")
  case False
  hence "n ∉ num ' set ps"
    using "1.prem" (3)[of n] <n ≤ ub> by auto
  hence [simp]: "ps' = ps"
    by (auto simp: ps'_def)
  show ?thesis using "1.prem" (3)[of p] p False
    by (cases "n = p") auto
  next
  case True
  hence "n ∈ num ' set ps"
    using "1.prem" (3)[of n] <n ≤ ub> by auto
  have *: "num p ≥ n" if "p ∈ set ps" for p
    using "1.prem" (3)[of "num p"] that <n ≤ ub>
    by (cases "num p ≤ ub") auto
  from <n ∈ num ' set ps> obtain x
    where ps_eq: "ps = x # ps'" "num x = n"
    using <sorted (map num ps)> <distinct (map num ps)> *
    by (cases ps) (fastforce simp: ps'_def)+
  show ?thesis
    by (cases "p = n")
      (use "1.prem" (3)[of p] p <distinct (map num ps)> in <auto
simp: ps_eq>)
  qed
next
have "rel (real_of_float acc') (S n)"
  by fact
also have "S n = S' (n + 1)"
  unfolding S_def S'_def by (simp add: lessThan_Suc_atMost)

```

```

    finally show "rel (real_of_float acc') (S' (n + 1))" .
  next
    show "check_aux ps' lb ub acc' (n + 1)"
      using "1.prem" (6,7)
      by (subst (asm) check_aux.simps) (auto simp: acc'_def ps'_def)
  next
    show "k ∈ {max lb (n+1)..ub}"
      using "1.prem" False by auto
  next
    show "rel (real_of_float (F prec x)) (f (num x))"
      if "x ∈ set ps'" for x
      using "1.prem" (4)[of x] that
      by (cases ps) (auto simp: ps'_def split: if_splits)
  qed (use <n ≤ ub> in <auto simp: acc'_def ps'_def>)
qed
qed

```

lemma check_correct:

```

  assumes "sorted (map num ps)" "distinct (map num ps)"
  assumes " $\bigwedge p. p \leq ub \implies p \in \text{num } \langle \text{set } ps \rangle \longleftrightarrow p \in A$ "
  assumes " $\bigwedge x. x \in \text{set } ps \implies \text{rel } (\text{real\_of\_float } (F \text{ prec } x)) (f (\text{num } x))$ "
  assumes "check ps lb ub"
  assumes "k ∈ {lb..ub}"
  shows "P k (S k)"
proof (rule check_aux_correct)
  define n where "n = (if ps = [] then lb else min lb (num (hd ps)))"
  have "n ≤ ub"
    using assms by (auto simp: n_def)
  show "sorted (map num ps)" "distinct (map num ps)"
    by fact+
  show "check_aux ps lb ub 0 n"
    using assms unfolding check_def n_def by simp
  show "k ∈ {max lb n..ub}"
    using assms by (auto simp: n_def)
  show "p ∈ num ⟨set ps⟩ ↔ p ∈ A ∧ n ≤ p" if "p ≤ ub" for p
    using assms (3)[of p] that <sorted (map num ps)>
    by (cases ps) (auto simp: n_def)

  have "A ∩ {..<n} = {}"
proof (intro equalityI subsetI)
  fix p assume p: "p ∈ A ∩ {..<n}"
  hence "p ∈ num ⟨set ps⟩"
    using assms (3)[of p] <n ≤ ub> by auto
  hence False
    using p <sorted (map num ps)> by (cases ps) (auto simp: n_def)
  thus "p ∈ {}" ..
qed auto
thus "rel (real_of_float 0) (S' n)"

```

```

    by (simp add: S'_def rel_0)
qed (use assms in auto)

```

```
end
```

```
end
```

1.1.4 The ϑ function

```
context
```

```
begin
```

```
interpretation primes_theta: chebyshev_check
```

```
  "λn. ln (real n)"
```

```
  "λprec n. the (lb_ln prec (Float (int n) 0))"
```

```
  "{p. prime p}"
```

```
  "float_plus_down"
```

```
  "(≤)"
```

```
  "λk x. x ≥ c * (real k + 1)"
```

```
  "λn. n"
```

```
  for c :: real
```

```
proof
```

```
  show "real_of_float (float_plus_down prec X Y) ≤ x + y"
```

```
    if "real_of_float X ≤ x" "real_of_float Y ≤ y"
```

```
    for x y :: real and X Y :: float and prec :: nat
```

```
    using that by (simp add: float_plus_down_le)
```

```
qed auto
```

```
definition check_theta_lower_aux
```

```
  where "check_theta_lower_aux = primes_theta.check_aux"
```

```
definition check_theta_lower where
```

```
  "check_theta_lower c prec lb ub =
```

```
    primes_theta.check c prec (primes_upto ub) lb ub"
```

```
lemma check_theta_lower_aux_code [code]:
```

```
  "check_theta_lower_aux c prec ps lb ub acc n =
```

```
    (if ub < n then True else let (acc', ps') =
```

```
      if ps ≠ [] ∧ hd ps = n
```

```
        then (float_plus_down prec acc (the (lb_ln prec (Float (int
```

```
(hd ps)) 0))), tl ps)
```

```
      else (acc, ps)
```

```
    in (n < lb ∨ c * (real n + 1) ≤ real_of_float acc') ∧
```

```
      check_theta_lower_aux c prec ps' lb
```

```
      ub acc' (n + 1))"
```

```
unfolding check_theta_lower_aux_def
```

```
by (rule primes_theta.check_aux.simps)
```

```

lemma check_theta_lower_code [code]:
  "check_theta_lower c prec lb ub = (let ps = primes_upto ub in
    check_theta_lower_aux c prec ps lb ub 0
    (if ps = [] then lb else min lb (hd ps)))"
  unfolding check_theta_lower_def primes_theta.check_def check_theta_lower_aux_def
  by (simp add: Let_def)

lemma check_theta_lower_correct:
  assumes "check_theta_lower c prec lb ub"
  shows "∀ x ∈ {real lb..real ub}. primes_theta x ≥ c * x"
proof
  fix x assume x: "x ∈ {real lb..real ub}"
  define k where "k = nat ⌊x⌋"
  show "c * x ≤ primes_theta x"
  proof (cases "c ≥ 0")
    case False
    hence "c * x ≤ 0"
      using x by (auto intro: mult_nonpos_nonneg)
    also have "0 ≤ primes_theta x"
      by (rule ∅_nonneg)
    finally show ?thesis .
  next
    case True
    hence "c * x ≤ c * (real k + 1)"
      using x by (intro mult_left_mono) (auto simp: k_def)
    also have "c * (real k + 1) ≤ primes_theta.S k"
    proof (rule primes_theta.check_correct)
      show "sorted (map (λn. n) (primes_upto ub))"
        "distinct (map (λn. n) (primes_upto ub))"
      by (simp_all add: sorted_primes_upto distinct_primes_upto)
      show "k ∈ {lb..ub}"
        using x by (auto simp: k_def le_nat_iff le_floor_iff nat_le_iff
        floor_le_iff)
      show "primes_theta.check c prec (primes_upto ub) lb ub"
        using assms by (simp add: check_theta_lower_def)
    next
      fix p assume "p ≤ ub"
      thus "p ∈ (λn. n) ' set (primes_upto ub) ↔ p ∈ {p. prime p}"
        by (auto simp: set_primes_upto)
    next
      fix n
      assume n: "n ∈ set (primes_upto ub)"
      hence "n > 0"
        by (auto simp: set_primes_upto prime_gt_0_nat)
      define x where "x = the (lb_ln prec (Float (int n) 0))"
      have "lb_ln prec (Float (int n) 0) ≠ None"
        using <n > 0> by (subst lb_ln.simps) auto
      hence "lb_ln prec (Float (int n) 0) = Some x"
        by (cases "lb_ln prec (Float (int n) 0)") (auto simp: x_def)
  end
end

```

```

    from lb_lnD[OF this] show "real_of_float x ≤ ln (real n)"
      by simp
  qed
  also have "primes_theta.S k = primes_theta k"
    unfolding primes_theta.S_def primes_theta_def prime_sum_upto_def
    by (intro sum.cong) auto
  also have "primes_theta k = primes_theta x"
    unfolding k_def by simp
  finally show "c * x ≤ primes_theta x" .
  qed
qed
end

```

```

context
begin

```

```

interpretation primes_theta: chebyshev_check
  "λn. ln (real n)"
  "λprec n. the (ub_ln prec (Float (int n) 0))"
  "{p. prime p}"
  "float_plus_up"
  "(≥)"
  "λk x. x ≤ c * real k"
  "λn. n"
  for c :: real
proof
  show "real_of_float (float_plus_up prec X Y) ≥ x + y"
    if "real_of_float X ≥ x" "real_of_float Y ≥ y"
    for x y :: real and X Y :: float and prec :: nat
    using that by (simp add: float_plus_up_le)
qed auto

```

```

definition check_theta_upper_aux
  where "check_theta_upper_aux = primes_theta.check_aux"

```

```

definition check_theta_upper where
  "check_theta_upper c prec lb ub =
    primes_theta.check c prec (primes_upto ub) lb ub"

```

```

lemma check_theta_upper_aux_code [code]:
  "check_theta_upper_aux c prec ps lb ub acc n =
    (if ub < n then True else let (acc', ps') =
      if ps ≠ [] ∧ hd ps = n
      then (float_plus_up prec acc (the (ub_ln prec (Float (int

```

```

(hd ps)) 0))), t1 ps)
      else (acc, ps)
      in (n < lb ∨ c * real n ≥ real_of_float acc') ∧
        check_theta_upper_aux c prec ps' lb
        ub acc' (n + 1))"
  unfolding check_theta_upper_aux_def
  by (rule primes_theta.check_aux.simps)

lemma check_theta_upper_code [code]:
  "check_theta_upper c prec lb ub = (let ps = primes_upto ub in
    check_theta_upper_aux c prec ps lb ub 0
    (if ps = [] then lb else min lb (hd ps)))"
  unfolding check_theta_upper_def primes_theta.check_def check_theta_upper_aux_def
  by (simp add: Let_def)

lemma check_theta_upper_correct:
  assumes "check_theta_upper c prec lb ub" "c ≥ 0"
  shows "∀x∈{real lb..real ub}. primes_theta x ≤ c * x"
proof
  fix x assume x: "x ∈ {real lb..real ub}"
  define k where "k = nat ⌊x⌋"
  have "primes_theta.S k ≤ c * real k"
  proof (rule primes_theta.check_correct)
    show "sorted (map (λn. n) (primes_upto ub))"
      "distinct (map (λn. n) (primes_upto ub))"
    by (simp_all add: sorted_primes_upto distinct_primes_upto)
    show "k ∈ {lb..ub}"
      using x by (auto simp: k_def le_nat_iff le_floor_iff nat_le_iff
        floor_le_iff)
    show "primes_theta.check c prec (primes_upto ub) lb ub"
      using assms by (simp add: check_theta_upper_def)
  next
    fix p assume "p ≤ ub"
    thus "p ∈ (λn. n) ` set (primes_upto ub) ↔ p ∈ {p. prime p}"
      by (auto simp: set_primes_upto)
  next
    fix n
    assume n: "n ∈ set (primes_upto ub)"
    hence "n > 0"
      by (auto simp: set_primes_upto prime_gt_0_nat)
    define x where "x = the (ub_ln prec (Float (int n) 0))"
    have "ub_ln prec (Float (int n) 0) ≠ None"
      using <n > 0> by (subst ub_ln.simps) auto
    hence "ub_ln prec (Float (int n) 0) = Some x"
      by (cases "ub_ln prec (Float (int n) 0)") (auto simp: x_def)
    from ub_lnD[OF this] show "real_of_float x ≥ ln (real n)"
      by simp
  qed
  also have "primes_theta.S k = primes_theta k"

```

```

    unfolding primes_theta.S_def primes_theta_def prime_sum_upto_def
    by (intro sum.cong) auto
  also have "primes_theta k = primes_theta x"
    unfolding k_def by simp
  also have "c * real k ≤ c * x"
    using <c ≥ 0> x by (intro mult_left_mono) (auto simp: k_def)
  finally show "primes_theta x ≤ c * x" .
qed

end

```

1.1.5 The ψ function

```

context
begin

```

```

interpretation primes_psi: chebyshev_check
  "λn. ln (real (aprime divisor n))"
  "λprec x. the (lb_ln prec (Float (int (snd x)) 0))"
  "{p. primepow p}"
  "float_plus_down"
  "(≤)"
  "λk x. x ≥ c * (real k + 1)"
  "fst"
  for c :: real
proof
  show "real_of_float (float_plus_down prec X Y) ≤ x + y"
    if "real_of_float X ≤ x" "real_of_float Y ≤ y"
    for x y :: real and X Y :: float and prec :: nat
    using that by (simp add: float_plus_down_le)
qed auto

```

```

definition check_psi_lower_aux
  where "check_psi_lower_aux = primes_psi.check_aux"

```

```

definition check_psi_lower where
  "check_psi_lower c prec lb ub =
    primes_psi.check c prec (prime_powers_upto ub) lb ub"

```

```

lemma check_psi_lower_aux_code [code]:
  "check_psi_lower_aux c prec ps lb ub acc n =
    (if ub < n then True else let (acc', ps') =
      if ps ≠ [] ∧ fst (hd ps) = n
      then (float_plus_down prec acc (the (lb_ln prec (Float (int
(snd (hd ps)) 0))), tl ps)
      else (acc, ps)
    in (n < lb ∨ c * (real n + 1) ≤ real_of_float acc') ∧
      check_psi_lower_aux c prec ps' lb

```

```

        ub acc' (n + 1))"
    unfolding check_psi_lower_aux_def
    by (rule primes_psi.check_aux.simps)

lemma check_psi_lower_code [code]:
  "check_psi_lower c prec lb ub = (let ps = prime_powers_upto ub in
    check_psi_lower_aux c prec ps lb ub 0
    (if ps = [] then lb else min lb (fst (hd ps))))"
  unfolding check_psi_lower_def primes_psi.check_def check_psi_lower_aux_def
  by (simp add: Let_def)

lemma check_psi_lower_correct:
  assumes "check_psi_lower c prec lb ub"
  shows "∀x∈{real lb..real ub}. primes_psi x ≥ c * x"
proof
  fix x assume x: "x ∈ {real lb..real ub}"
  define k where "k = nat ⌊x⌋"
  show "c * x ≤ primes_psi x"
  proof (cases "c ≥ 0")
    case False
    hence "c * x ≤ 0"
    using x by (auto intro: mult_nonpos_nonneg)
    also have "0 ≤ primes_psi x"
    by (rule ψ_nonneg)
    finally show ?thesis .
  next
    case True
    hence "c * x ≤ c * (real k + 1)"
    using x by (intro mult_left_mono) (auto simp: k_def)
    also have "c * (real k + 1) ≤ primes_psi.S k"
    proof (rule primes_psi.check_correct)
      show "sorted (map fst (prime_powers_upto ub))"
      "distinct (map fst (prime_powers_upto ub))"
      by (simp_all add: sorted_prime_powers_upto distinct_prime_powers_upto)
      show "k ∈ {lb..ub}"
      using x by (auto simp: k_def le_nat_iff le_floor_iff nat_le_iff
      floor_le_iff)
      show "primes_psi.check c prec (prime_powers_upto ub) lb ub"
      using assms by (simp add: check_psi_lower_def)
    next
      fix p assume "p ≤ ub"
      thus "p ∈ fst ' set (prime_powers_upto ub) ⟷ p ∈ {p. primepow
      p}"
      by (force simp: set_prime_powers_upto)
    next
      fix y
      assume y: "y ∈ set (prime_powers_upto ub)"
      hence "snd y > 0"
      by (auto simp: set_prime_powers_upto intro!: aprimedivisor_pos_nat

```



```

primepow_gt_Suc_0)
  define x where "x = the (lb_ln prec (Float (int (snd y)) 0))"
  have "lb_ln prec (Float (int (snd y)) 0) ≠ None"
    using <snd y > 0> by (subst lb_ln.simps) auto
  hence "lb_ln prec (Float (int (snd y)) 0) = Some x"
    by (cases "lb_ln prec (Float (int (snd y)) 0)") (auto simp: x_def)
  from lb_lnD[OF this] show "real_of_float x ≤ ln (real (aprime divisor
(fst y)))"
    using y by (auto simp: set_prime_powers_upto)
  qed
  also have "primes_psi.S k = primes_psi k"
    unfolding primes_psi.S_def primes_psi_def sum_upto_def
    by (intro sum.mono_neutral_cong_left) (auto simp: primepow_gt_0_nat
mangoldt_def)
  also have "primes_psi k = primes_psi x"
    unfolding k_def by simp
  finally show "c * x ≤ primes_psi x" .
  qed
qed
end

```

```

context
begin

```

```

interpretation primes_psi: chebyshev_check
  "λn. ln (real (aprime divisor n))"
  "λprec x. the (ub_ln prec (Float (int (snd x)) 0))"
  "{p. primepow p}"
  "float_plus_up"
  "(≥)"
  "λk x. x ≤ c * real k"
  "fst"
  for c :: real
proof
  show "real_of_float (float_plus_up prec X Y) ≥ x + y"
    if "real_of_float X ≥ x" "real_of_float Y ≥ y"
    for x y :: real and X Y :: float and prec :: nat
    using that by (simp add: float_plus_up_le)
qed auto

```

```

definition check_psi_upper_aux
  where "check_psi_upper_aux = primes_psi.check_aux"

```

```

definition check_psi_upper where
  "check_psi_upper c prec lb ub =
    primes_psi.check c prec (prime_powers_upto ub) lb ub"

```

```

lemma check_psi_upper_aux_code [code]:
  "check_psi_upper_aux c prec ps lb ub acc n =
    (if ub < n then True else let (acc', ps') =
      if ps ≠ [] ∧ fst (hd ps) = n
      then (float_plus_up prec acc (the (ub_ln prec (Float (int
(snd (hd ps))) 0))), tl ps)
      else (acc, ps)
    in (n < lb ∨ c * real n ≥ real_of_float acc') ∧
      check_psi_upper_aux c prec ps' lb
      ub acc' (n + 1))"
  unfolding check_psi_upper_aux_def
  by (rule primes_psi.check_aux.simps)

lemma check_psi_upper_code [code]:
  "check_psi_upper c prec lb ub = (let ps = prime_powers_upto ub in
    check_psi_upper_aux c prec ps lb ub 0
    (if ps = [] then lb else min lb (fst (hd ps))))"
  unfolding check_psi_upper_def primes_psi.check_def check_psi_upper_aux_def
  by (simp add: Let_def)

lemma check_psi_upper_correct:
  assumes "check_psi_upper c prec lb ub" "c ≥ 0"
  shows "∀x∈{real lb..real ub}. primes_psi x ≤ c * x"
proof
  fix x assume x: "x ∈ {real lb..real ub}"
  define k where "k = nat ⌊x⌋"
  have "primes_psi.S k ≤ c * real k"
  proof (rule primes_psi.check_correct)
    show "sorted (map fst (prime_powers_upto ub))"
      "distinct (map fst (prime_powers_upto ub))"
    by (simp_all add: sorted_prime_powers_upto distinct_prime_powers_upto)
    show "k ∈ {lb..ub}"
      using x by (auto simp: k_def le_nat_iff le_floor_iff nat_le_iff
floor_le_iff)
    show "primes_psi.check c prec (prime_powers_upto ub) lb ub"
      using assms by (simp add: check_psi_upper_def)
  next
    fix p assume "p ≤ ub"
    thus "p ∈ fst ` set (prime_powers_upto ub) ⟷ p ∈ {p. primepow p}"
      by (force simp: set_prime_powers_upto)
  next
    fix y
    assume y: "y ∈ set (prime_powers_upto ub)"
    hence "snd y > 0"
      by (auto simp: set_prime_powers_upto intro!: aprimedivisor_pos_nat
primepow_gt_Suc_0)
    define x where "x = the (ub_ln prec (Float (int (snd y)) 0))"
    have "ub_ln prec (Float (int (snd y)) 0) ≠ None"

```

```

    using <snd y > 0> by (subst ub_ln.simps) auto
    hence "ub_ln prec (Float (int (snd y)) 0) = Some x"
      by (cases "ub_ln prec (Float (int (snd y)) 0)") (auto simp: x_def)
    from ub_lnD[OF this] show "real_of_float x ≥ ln (real (aprime divisor
(fst y)))"
      using y by (auto simp: set_prime_powers_upto)
  qed
  also have "primes_psi.S k = primes_psi k"
    unfolding primes_psi.S_def primes_psi_def sum_upto_def
    by (intro sum.mono_neutral_cong_left) (auto simp: primepow_gt_0_nat
mangoldt_def)
  also have "primes_psi k = primes_psi x"
    unfolding k_def by simp
  also have "c * real k ≤ c * x"
    using x assms by (intro mult_left_mono) (auto simp: k_def)
  finally show "primes_psi x ≤ c * x" .
end

```

```

end
theory Chebyshev_Prime_Bounds
imports
  "Prime_Number_Theorem.Prime_Counting_Functions"
  "Prime_Distribution_Elementary.Prime_Distribution_Elementary_Library"
  "Prime_Distribution_Elementary.Primorial"
  "HOL-Decision_Procs.Approximation"
  "HOL-Library.Code_Target_Natural"
  Chebyshev_Prime_Exhaust
begin

```

1.2 Auxiliary material

```

context comm_monoid_set
begin

lemma union_disjoint':
  assumes "finite C" "A ∪ B = C" "A ∩ B = {}"
  shows "f (F g A) (F g B) = F g C"
  using union_disjoint[of A B g] assms by auto

end

lemma sum_mset_nonneg:
  fixes X :: "'a :: ordered_comm_monoid_add multiset"
  shows "(∧x. x ∈# X ⇒ x ≥ 0) ⇒ sum_mset X ≥ 0"
  by (induction X) (auto)

lemma of_int_sum_mset: "of_int (sum_mset M) = sum_mset (image_mset of_int

```

```

M)"
  by (induction M) auto

lemma sum_sum_mset: "( $\sum_{x \in A}. \sum_{y \in B}. f\ x\ y$ ) = ( $\sum_{y \in B}. \sum_{x \in A}. f\ x\ y$ )"
  by (induction B) (auto simp: algebra_simps sum.distrib)

lemma sum_mset_diff_distrib:
  fixes f g :: "'a  $\Rightarrow$  'b :: ab_group_add"
  shows "( $\sum_{x \in \#A}. f\ x - g\ x$ ) = ( $\sum_{x \in \#A}. f\ x$ ) - ( $\sum_{x \in \#A}. g\ x$ )"
  by (induction A) (auto simp: algebra_simps)

lemma sum_mset_neg_distrib:
  fixes f :: "'a  $\Rightarrow$  'b :: ab_group_add"
  shows "( $\sum_{x \in \#A}. -f\ x$ ) = -( $\sum_{x \in \#A}. f\ x$ )"
  by (induction A) (auto simp: algebra_simps)

```

1.3 Bounds for the remainder in Stirling's approximation

definition ln_fact_remainder :: "real \Rightarrow real" where

```
"ln_fact_remainder x = ln (fact (nat [x])) - (x * ln x - x)"
```

lemma ln_fact_remainder_bounds:

```

assumes x: "x  $\geq$  3"
shows "ln_fact_remainder x  $\leq$  ln x / 2 + ln (2 * pi) / 2 + 1 / (12 * [x])"
  and "ln_fact_remainder x  $\geq$  -ln x / 2 + ln (2 * pi) / 2 - 1 / (2 * x)"

```

proof -

```
define n where "n = nat [x]"
```

```
define f where "f = ( $\lambda t. t * (ln t - 1) + ln t / 2$ ) :: real"
```

```
have "ln [x]  $\geq$  1"
```

proof -

```
have "1  $\leq$  ln (3 :: real)"
```

```
by (approximation 10)
```

```
also have "ln 3  $\leq$  ln [x]"
```

```
using assms by simp
```

```
finally show ?thesis .
```

qed

```
have n: "n  $\geq$  1"
```

```
using x by (auto simp: n_def le_nat_iff)
```

```
have "ln_fact_remainder x = ln (fact n) + ln x / 2 - f x"
```

```
by (simp add: ln_fact_remainder_def n_def f_def algebra_simps)
```

```
also have "ln (fact n)  $\leq$  ln (2 * pi) / 2 + f n + 1 / (12 * n)"
```

```
using ln_fact_bounds(2)[of n] n by (auto simp: f_def ln_mult add_divide_distrib algebra_simps)
```

```
also have "... + ln x / 2 - f x = ln x / 2 + (f n - f x) + ln (2 * pi)
```

```

/ 2 + 1 / (12 * n)"
  using n by (simp add: algebra_simps ln_mult)
also have "f n ≤ f x"
  unfolding f_def using assms <ln [x] ≥ 1>
  by (intro add_mono mult_mono) (auto simp: n_def)
finally show "ln_fact_remainder x ≤ ln x / 2 + ln (2 * pi) / 2 + 1 /
(12 * [x])"
  using assms by (simp add: n_def)

define f' :: "real ⇒ real" where "f' = (λx. ln x + 1 / (2 * x))"
have f'_mono: "f' x ≤ f' y" if "x ≤ y" "x ≥ 1 / 2" for x y :: real
  using that(1)
proof (rule DERIV_nonneg_imp_nondecreasing)
  fix t assume t: "t ≥ x" "t ≤ y"
  hence "t > 0"
    using <x ≥ 1 / 2> by auto
  have "(t - 1 / 2) / t ^ 2 ≥ 0"
    using t that by auto
  have "(f' has_field_derivative (1 / t - 1 / (2 * t ^ 2))) (at t)"
    using <t > 0> by (auto simp: f'_def power2_eq_square intro!: derivative_eq_intros)
  also have "1 / t - 1 / (2 * t ^ 2) = (t - 1 / 2) / t ^ 2"
    using <t > 0> by (simp add: field_simps eval_nat_numeral del: div_diff)
  finally show "∃y. (f' has_real_derivative y) (at t) ∧ 0 ≤ y"
    using <(t - 1 / 2) / t ^ 2 ≥ 0> by blast
qed

have f'_nonneg: "f' t ≥ 0" if "t ≥ 3" for t
proof -
  have "0 ≤ f' 3"
    unfolding f'_def by (approximation 10)
  also have "f' 3 ≤ f' t"
    by (rule f'_mono) (use that in auto)
  finally show ?thesis .
qed

have "f x - f n ≤ f' x * frac x"
proof (cases "n < x")
  case False
  hence "x = n"
    using assms unfolding n_def by linarith
  thus ?thesis using f'_nonneg[of x] assms
    by (simp add: n_def)
next
  case True
  have "∃z::real. z > n ∧ z < x ∧ f x - f n = (x - n) * f' z"
    using True assms n
    by (intro MVT2) (auto intro!: derivative_eq_intros simp: f_def f'_def)
  then obtain z :: real where z: "z > n" "z < x" "f x - f n = (x -
n) * f' z"

```

```

    by blast
  have "f' z ≤ f' x"
    by (rule f'_mono) (use z assms n in auto)

  have "f x - f n = (x - n) * f' z"
    by fact
  also have "... ≤ (x - n) * f' x"
    using <f' z ≤ f' x> True by (intro mult_left_mono) auto
  also have "x - n = frac x"
    using assms by (simp add: n_def frac_def)
  finally show ?thesis
    by (simp add: mult_ac)
qed

also have "... ≤ f' x * 1"
  using frac_lt_1[of x] f'_nonneg[of x] assms
  by (intro mult_left_mono) auto
finally have "f n - f x ≥ -1 / (2 * x) - ln x"
  by (simp add: f'_def)

have "-ln x / 2 - 1 / (2 * x) + ln (2 * pi) / 2 =
  ln x / 2 + (-1 / (2 * x) - ln x) + ln (2 * pi) / 2"
  by (simp add: algebra_simps)
also have "-1 / (2 * x) - ln x ≤ f n - f x"
  by fact
also have "ln x / 2 + (f n - f x) + ln (2 * pi) / 2 =
  ln (2 * pi) / 2 + f n + ln x / 2 - f x"
  by (simp add: algebra_simps)
also have "ln (2 * pi) / 2 + f n ≤ ln (fact n)"
  using ln_fact_bounds(1)[of n] n by (auto simp: f_def ln_mult add_divide_distrib
algebra_simps)
also have "ln (fact n) + ln x / 2 - f x = ln_fact_remainder x"
  by (simp add: ln_fact_remainder_def f_def n_def algebra_simps)
finally show "ln_fact_remainder x ≥ -ln x / 2 + ln (2 * pi) / 2 - 1
/ (2 * x)"
  by simp
qed

lemma abs_ln_fact_remainder_bounds:
  assumes x: "x ≥ 3"
  shows "|ln_fact_remainder x| < ln x / 2 + 1"
proof -
  have "ln_fact_remainder x ≤ ln x / 2 + (ln (2 * pi) / 2 + 1 / (12 *
[x]))"
    using ln_fact_remainder_bounds(1)[of x] assms by (simp add: algebra_simps)
  also have "1 / (12 * [x]) ≤ 1 / 36"
    using assms by auto
  also have "ln (2 * pi) / 2 + 1 / 36 < 1"
    by (approximation 10)

```

```

finally have less: "ln_fact_remainder x < ln x / 2 + 1"
  by simp

have "-(ln x / 2 + 1) = -ln x / 2 + (-1)"
  by simp
also have "-1 < 0 - 1 / (2 * x)"
  using assms by simp
also have "0 ≤ ln (2 * pi) / 2"
  using pi_gt3 by simp
also have "-ln x / 2 + (ln (2 * pi) / 2 - 1 / (2 * x)) ≤ ln_fact_remainder
x"
  using ln_fact_remainder_bounds(2)[of x] assms by (simp add: algebra_simps)
finally have "- (ln x / 2 + 1) < ln_fact_remainder x" by - simp_all
with less show ?thesis
  by linarith
qed

```

1.4 Approximating ψ

unbundle prime_counting_notation

lemma primes_psi_lower_rec:

```

fixes f :: "real ⇒ real"
assumes "\x. x ≥ x0 ⇒ f x ≤ f (x / c) + h x"
assumes "x0 > 0" "x * c ≥ x0 * c ^ n" "c ≥ 1"
shows "f x ≤ f (x / c ^ n) + (∑ k<n. h (x / c ^ k))"
using assms(2-)
proof (induction n arbitrary: x)
  case 0
  thus ?case by auto
next
  case (Suc n)
  have "0 < x0 * c ^ n"
    using Suc.prem1 by auto
  also have "... ≤ x"
    using Suc.prem2 by auto
  finally have "x > 0" .

  have "x0 * c ^ n ≤ 1 * x"
    using Suc.prem1 by simp
  also have "1 * x ≤ c * x"
    by (rule mult_right_mono) (use Suc.prem1 <x > 0> in auto)
  finally have "f x ≤ f (x / c ^ n) + (∑ k<n. h (x / c ^ k))"
    by (intro Suc.IH) (use Suc.prem1 in <auto simp: mult_ac>)
  also have "f (x / c ^ n) ≤ f (x / c ^ n / c) + h (x / c ^ n)"
    by (rule assms(1)) (use Suc.prem1 <x > 0> in <auto simp: field_simps
less_imp_le>)
  finally show ?case
    by (simp add: mult_ac add_ac)

```

qed

```
locale chebyshev_multiset =
  fixes L :: "int multiset"
  assumes L_nonzero: "0  $\notin$  # L"
begin
```

```
definition chi_L :: "real  $\Rightarrow$  int" (" $\chi_L$ ")
  where "chi_L t = ( $\sum_{l \in \#L}. \text{sgn } l * \lfloor t / |l| \rfloor$ )"
```

```
definition psi_L :: "real  $\Rightarrow$  real" (" $\psi_L$ ")
  where "psi_L x = sum_upto ( $\lambda d. \text{mangoldt } d * \text{chi}_L (x / d)$ ) x"
```

```
definition alpha_L :: real (" $\alpha_L$ ")
  where "alpha_L = -( $\sum_{l \in \#L}. \ln |l| / l$ )"
```

```
definition period :: nat
  where "period = nat (Lcm (set_mset L))"
```

```
lemma period_pos: "period > 0"
```

```
proof -
  have "Lcm (set_mset L)  $\neq$  0"
    using L_nonzero unfolding period_def by (subst Lcm_0_iff) auto
  moreover have "Lcm (set_mset L)  $\geq$  0"
    by auto
  ultimately have "Lcm (set_mset L) > 0"
    by linarith
  thus ?thesis
    by (simp add: period_def)
```

qed

```
lemma dvd_period: "l  $\in$  # L  $\implies$  l dvd period"
  unfolding period_def by auto
```

```
lemma chi_L_decompose:
```

```
" $\chi_L (x + \text{of\_int } (m * \text{int period})) = \chi_L x + m * \text{int period} * (\sum_{l \in \#L}. \lfloor x / |l| \rfloor)$ "
```

```
proof -
```

```
  have "real_of_int ( $\chi_L (x + \text{of\_int } (m * \text{int period}))$ ) =
    ( $\sum_{l \in \#L}. \text{of\_int } (\text{sgn } l * \lfloor (x + \text{of\_int } m * \text{real period}) / \text{real\_of\_int } |l| \rfloor$ )"
```

```
    by (simp add: chi_L_def of_int_sum_mset multiset.map_comp o_def)
  also have "... = ( $\sum_{l \in \#L}. \text{real\_of\_int } (\text{sgn } l * (\lfloor x / \text{of\_int } |l| \rfloor)) + m * \text{period} / l$ )"
```

```
  proof (intro arg_cong[of _ _ sum_mset] image_mset_cong, goal_cases)
    case (1 l)
    with L_nonzero have [simp]: "l  $\neq$  0"
      by auto
```



```

    have "(x + of_int m * real period) / real_of_int |l| =
      x / of_int |l| + of_int (m * period div |l|)"
      using dvd_period[of l] 1 by (subst real_of_int_div) (auto simp:
field_simps)
    also have "floor ... = ⌊x / of_int |l| :: real⌋ + m * period div |l|"
      by (subst floor_add_int) auto
    also have "real_of_int ... = ⌊x / of_int |l|⌋ + m * period / |l|"
      using dvd_period[of l] 1 by (simp add: real_of_int_div)
    also have "sgn l * ... = sgn l * ⌊x / of_int |l|⌋ + m * period / l"
      by (simp add: sgn_if)
    finally show ?case
      by simp
  qed
  also have "... = of_int (χL x) + (∑ l∈#L. m * period / l)"
    by (subst sum_mset.distrib)
      (auto simp: chi_L_def of_int_sum_mset multiset.map_comp o_def)
  also have "(∑ l∈#L. m * period / l) = m * period * (∑ l∈#L. 1 / l)"
    by (simp add: sum_mset_distrib_left)
  finally show ?thesis
    by simp
qed

```

```

lemma chi_L_floor: "chi_L (floor x) = chi_L x"
  unfolding chi_L_def
proof (intro arg_cong[of _ _ sum_mset] image_mset_cong, goal_cases)
  case (1 l)
  thus ?case
    using floor_divide_real_eq_div[of "|l|" x] floor_divide_of_int_eq[of
"x|" "|l|"]
    by auto
qed
end

```

```

locale balanced_chebyshev_multiset = chebyshev_multiset +
  assumes balanced: "(∑ l∈#L. 1 / l) = 0"
begin

```

```

lemma chi_L_mod: "χL (of_int (a mod int period)) = χL (of_int a)"
proof -
  have a: "a = a mod period + period * (a div period)"
    by simp
  have "of_int a = real_of_int (a mod int period) +
    real_of_int (a div int period * int period)"
    by (subst a, unfold of_int_add) auto
  also have "real_of_int (χL ...) = real_of_int (χL (real_of_int (a mod
int period)))"
    using balanced by (subst chi_L_decompose) auto

```

```

    finally show ?thesis
      by linarith
qed

```

```

sublocale chi: periodic_fun_simple chi_L "of_int period"
proof
  fix x :: real
  have " $\chi_L (x + \text{real\_of\_int } (\text{int period})) = \chi_L (\text{of\_int } (\lfloor x + \text{real\_of\_int } (\text{int period}) \rfloor \text{ mod int period}))$ "
    unfolding chi_L_mod chi_L_floor ..
  also have " $\lfloor x + \text{real\_of\_int } (\text{int period}) \rfloor \text{ mod int period} = \lfloor x \rfloor \text{ mod int period}$ "
    by simp
  also have " $\chi_L \dots = \chi_L x$ "
    by (simp add: chi_L_mod chi_L_floor)
  finally show " $\chi_L (x + \text{real\_of\_int } (\text{int period})) = \chi_L x$ " .
qed

```

definition psi_L_remainder where

```

  "psi_L_remainder x = ( $\sum_{l \in \#L. \text{sgn } l * \text{ln\_fact\_remainder } (x / |l|)$ )"

```

lemma abs_sum_mset_le:

```

  fixes f :: "'a  $\Rightarrow$  'b :: ordered_ab_group_add_abs"
  shows " $|\sum_{x \in \#A. f x}| \leq (\sum_{x \in \#A. |f x|)$ "
  by (induction A) (auto intro: order.trans[OF abs_triangle_ineq])

```

lemma psi_L_remainder_bounds:

```

  fixes x :: real
  assumes x: " $x \geq 3$ " " $\bigwedge l. l \in \#L \implies x \geq 3 * |l|$ "
  shows " $|\text{psi\_L\_remainder } x| \leq \ln x * \text{size } L / 2 - 1/2 * (\sum_{l \in \#L. \ln |l|) + \text{size } L$ "

```

proof -

```

  have nonzero: " $l \neq 0$ " if " $l \in \#L$ " for l
    using L_nonzero that by auto
  have "psi_L_remainder x = ( $\sum_{l \in \#L. \text{sgn } l * \text{ln\_fact\_remainder } (x / |l|)$ )"
    by (simp add: psi_L_remainder_def)
  also have " $|\dots| \leq (\sum_{l \in \#L. |\text{sgn } l * \text{ln\_fact\_remainder } (x / |l|)|)$ "
    by (rule abs_sum_mset_le)
  also have " $\dots = (\sum_{l \in \#L. |\text{ln\_fact\_remainder } (x / |l|)|)$ "
    by (intro arg_cong[of _ _ sum_mset] image_mset_cong)
      (auto simp: nonzero abs_mult simp flip: of_int_abs)
  also have " $\dots \leq (\sum_{l \in \#L. \ln (x / |l|) / 2 + 1)$ "
    using x
    by (intro sum_mset_mono less_imp_le[OF abs_ln_fact_remainder_bounds])
      (auto simp: nonzero field_simps)
  also have " $\dots = (\sum_{l \in \#L. 1 / 2 * (\ln x - \ln |l|) + 1)$ "
    using assms
    by (intro arg_cong[of _ _ sum_mset] image_mset_cong) (auto simp: algebra_simps)

```

```

ln_div nonzero)
  also have "... = ln x / 2 * size L + (-1/2) * ( $\sum_{l \in \#L}. \ln |l|$ ) + size
  L"
  unfolding sum_mset_distrib_left of_int_sum_mset
  by (simp add: sum_mset.distrib sum_mset_diff_distrib diff_divide_distrib
sum_mset_neg_distrib)
  finally show ?thesis
  using assms by (simp add: mult_left_mono divide_right_mono add_mono)
qed

lemma psi_L_eq:
  assumes "x > 0"
  shows "psi_L x =  $\alpha_L * x + \text{psi\_L\_remainder } x$ "
proof -
  have "psi_L x = ( $\sum_{l \in \#L}. \text{sgn } l * \text{sum\_upto } (\lambda d. \text{mangoldt } d * \lfloor x / (d * |l|) \rfloor) x$ )"
  by (simp add: psi_L_def chi_L_def sum_upto_def sum_mset_distrib_left
of_int_sum_mset
multiset.map_comp o_def sum_sum_mset algebra_simps sum_distrib_left
sum_distrib_right)
  also have "... = ( $\sum_{l \in \#L}. \text{sgn } l * \text{sum\_upto } (\lambda d. \text{mangoldt } d * \lfloor x / (d * |l|) \rfloor) (x / |l|)$ )"
  proof (intro arg_cong[of _ _ sum_mset] image_mset_cong, goal_cases)
    case (1 l)
    have "l  $\neq$  0"
    using 1 L_nonzero by auto

    have "sum_upto ( $\lambda d. \text{mangoldt } d * \text{real\_of\_int } \lfloor x / \text{real\_of\_int } (\text{int }
d * |l|) \rfloor) (x / \text{real\_of\_int } |l|) = \text{sum\_upto } (\lambda d. \text{mangoldt } d * \text{real\_of\_int } \lfloor x / \text{real\_of\_int } (\text{int }
d * |l|) \rfloor) x"$ )"
    unfolding sum_upto_def
  proof (intro sum.mono_neutral_left subsetI ballI, goal_cases)
    case (2 d)
    hence "real d  $\leq$  x / |real_of_int l|"
    by auto
    also have "...  $\leq$  x / 1"
    using <l  $\neq$  0> and assms by (intro divide_left_mono) auto
    finally show ?case
    using 2 by auto
  next
    case (3 d)
    hence "x < d * |l|" and "d > 0"
    using <l  $\neq$  0> and assms by (auto simp: field_simps)
    hence "x / real_of_int (int d * |l|)  $\geq$  0" and "x / real_of_int
(int d * |l|) < 1"
    using assms by auto
    hence " $\lfloor x / \text{real\_of\_int } (\text{int } d * |l|) \rfloor = 0$ "
    by linarith
  end
end

```

```

    thus ?case
      by simp
  qed auto
  thus ?case
    by simp
qed

also have "... = (∑ l∈#L. sgn l * ln (fact (nat ⌊x/|l|⌋)))"
  by (subst ln_fact_conv_sum_mangoldt [symmetric]) (auto simp: mult_ac)

also have "... = (∑ l∈#L. x / l * ln x - x * ln |l| / l - x / l + sgn
l * ln_fact_remainder (x / |l|))"
  proof (intro arg_cong[of _ _ sum_mset] image_mset_cong, goal_cases)
    case (1 l)
      hence [simp]: "l ≠ 0"
        using L_nonzero by auto
      have "ln (fact (nat ⌊x/|l|⌋)) = x / |l| * ln (x / |l|) - x / |l| + ln_fact_remainder
(x / |l|)"
        by (simp add: ln_fact_remainder_def)
      also have "real_of_int (sgn l) * ... = x / l * ln x - x * ln |l| /
l - x / l + sgn l * ln_fact_remainder (x / |l|)"
        using assms by (auto simp: sgn_if ln_div diff_divide_distrib algebra_simps)
      finally show ?case .
    qed
  also have "... = (x * ln x - x) * (∑ l∈#L. 1 / l) - x * (∑ l∈#L. ln
|l| / l) + (∑ l∈#L. sgn l * ln_fact_remainder (x / |l|))"
    by (simp add: sum_mset.distrib sum_mset_diff_distrib sum_mset_distrib_left
diff_divide_distrib)
  also have "... = αL * x + psi_L_remainder x"
    by (subst balanced) (auto simp: alpha_L_def psi_L_remainder_def)
  finally show ?thesis .
qed

lemma primes_psi_lower_bound:
  fixes x C :: real
  defines "x0 ≡ Max (insert 3 ((λl. 3 * |l|) ` set_mset L))"
  assumes x: "x ≥ x0"
  assumes chi_le1: "∧n. n ∈ {0..<period>} ⇒ χL (real n) ≤ 1"
  defines "C ≡ 1 / 2 * (∑ l∈#L. ln |l|) - size L"
  shows "ψ x ≥ αL * x - ln x * size L / 2 + C"
proof -
  have chi_le1': "χL x ≤ 1" for x
  proof -
    have "χL x = χL (floor x mod period)"
      by (simp add: chi_L_mod chi_L_floor)
    also have "floor x mod period = real (nat (floor x mod period))"
      using period_pos by auto
    also have "χL ... ≤ 1"
      by (rule chi_le1) (use period_pos in <auto simp: nat_less_iff>)
  qed

```

```

    finally show ?thesis .
qed

have x0: "x0 ≥ 3" "\^1. l ∈# L ⇒ x0 ≥ 3 * |l|"
  unfolding x0_def by auto

have *: "x * y ≤ x" if "y ≤ 1" "x ≥ 0" for x y :: real
  using mult_left_mono[OF that] by auto

have "|psi_L_remainder x| ≤ ln x * real (size L) / 2 -
      1 / 2 * (∑ l ∈# L. ln (real_of_int |l|)) + real (size L)"
  by (rule psi_L_remainder_bounds)
  (use x x0 in <force simp flip: of_int_abs>)+
hence "|psi_L_remainder x| ≤ ln x * size L / 2 - C"
  by (simp add: C_def algebra_simps)
hence "α_L * x - ln x * size L / 2 + C ≤ α_L * x + psi_L_remainder x"
  by linarith
also have "α_L * x + psi_L_remainder x = ψ_L x"
  using x x0(1) by (subst psi_L_eq) auto
also have "ψ_L x ≤ ψ x"
  unfolding psi_L_def primes_psi_def sum_upto_def
  by (intro sum_mono *) (auto simp: mangoldt_nonneg chi_le1')
finally show ?thesis
  by (simp add: C_def)
qed

end

lemma psi_lower_bound_precise:
  assumes x: "x ≥ 90"
  shows "ψ x ≥ 0.92128 * x - 2.5 * ln x - 1.6"
proof -
  interpret balanced_chebyshev_multiset "{#1, -2, -3, -5, 30#}"
  by unfold_locales auto

  define C :: real where "C = ((ln 2 + (ln 3 + (ln 5 + ln 30))) / 2 -
5)"
  have "alpha_L = ln 2 / 2 - (ln 30 / 30 - ln 5 / 5 - ln 3 / 3)"
    by (simp add: alpha_L_def)
  also have "... ≥ 0.92128"
    by (approximation 30)
  finally have "alpha_L ≥ 0.92128" .
  have "C ≥ -1.6"
    unfolding C_def by (approximation 20)

  have "0.92128 * x - ln x * 5 / 2 + (-1.6) ≤ alpha_L * x - ln x * 5
/ 2 + C"
  using <alpha_L ≥ _> <C ≥ _> x by (intro diff_mono add_mono mult_right_mono)
auto

```

```

also have "chi_L k ≤ 1" if "k ∈ {..<30}" for k :: nat
  using that unfolding lessThan_nat_numeral pred_numeral_simps arith_simps
  by (elim insertE) (auto simp: chi_L_def)
hence "alpha_L * x - ln x * 5 / 2 + C ≤ ψ x"
  using primes_psi_lower_bound[of x] x by (simp add: C_def period_def)
finally show ?thesis
  by (simp add: mult_ac)
qed

```

```

context balanced_chebyshev_multiset
begin

```

```

lemma psi_upper_bound:

```

```

  fixes x c C :: real
  defines "x0 ≡ Max ({3, 55 * c} ∪ {3 * |l| |l. l ∈# L})"
  assumes x: "x ≥ x0"
  assumes chi_nonneg: "∧n. n ∈ {0..<period} ⇒ χ_L (real n) ≥ 0"
  assumes chi_ge1: "∧n. real n ∈ {1..<c} ⇒ χ_L (real n) ≥ 1"
  assumes c: "c > 1" "c ≤ period"
  assumes "α_L ≥ 0"
  shows "ψ x ≤ c / (c - 1) * α_L * x + (3 * size L) / (4 * ln c) * ln
x ^ 2 + ψ x0"

```

```

proof -

```

```

  have L_nonzero': "l ≠ 0" if "l ∈# L" for l
    using that L_nonzero by auto

```

```

  have chi_nonneg: "χ_L x ≥ 0" for x

```

```

  proof -

```

```

    have "χ_L x = χ_L (floor x mod period)"

```

```

      by (simp add: chi_L_mod chi_L_floor)

```

```

    also have "floor x mod period = real (nat (floor x mod period))"

```

```

      using period_pos by auto

```

```

    also have "χ_L ... ≥ 0"

```

```

      by (rule chi_nonneg) (use period_pos in <auto simp: nat_less_iff>)

```

```

    finally show ?thesis .

```

```

  qed

```

```

  have chi_ge1: "χ_L x ≥ 1" if "x ≥ 1" "x < c" for x

```

```

  proof -

```

```

    have "χ_L x = χ_L (floor x mod period)"

```

```

      by (simp add: chi_L_mod chi_L_floor)

```

```

    also have "floor x mod period = real (nat (floor x mod period))"

```

```

      using period_pos by auto

```

```

    also have "χ_L ... ≥ 1"

```

```

  proof (rule chi_ge1)

```

```

    have "real_of_int ⌊x⌋ < c"

```

```

      using that by linarith

```

```

    hence "real_of_int (⌊x⌋ mod int period) < c"

```

```

      using that period_pos c by simp

```

```

    moreover have "1 ≤ ⌊x⌋ mod int period"
      by (use period_pos c that in <auto simp: floor_less_iff>)
    ultimately show "real (nat (⌊x⌋ mod int period)) ∈ {1..<c>}"
      by auto
  qed
  finally show ?thesis .
qed

have "finite {3 * l | l. l ∈# L}"
  by auto
have x1: "x0 ≥ 3" "x0 ≥ 55 * c"
  unfolding x0_def by (rule Max_ge; simp)+
have x2: "3 * |l| ≤ x0" if "l ∈# L" for l
  unfolding x0_def by (rule Max_ge) (use that in auto)

define C where "C = 1/2 * (∑ l ∈# L. ln |l|) - size L"
have *: "x ≤ x * y" if "y ≥ 1" "x ≥ 0" for x y :: real
  using mult_left_mono[of 1 y x] that by simp

have rec: "ψ x ≤ ψ (x / c) + α_L * x + ln x * size L / 2 - C" if x:
"x ≥ x0" for x :: real
proof -
  have "x / c ≤ x"
    using c using divide_left_mono[of 1 c x] <x0 ≥ 3> x by auto
  have "ψ x = ψ (x / c) + (∑ d | d > 0 ∧ real d ∈ {x/c<..x}. mangoldt
d)"
    unfolding ψ_def sum_upto_def
    by (rule sum.union_disjoint' [symmetric])
      (use c <x / c ≤ x> in auto)
  also have "(∑ d | d > 0 ∧ real d ∈ {x/c<..x}. mangoldt d) ≤
(∑ d | d > 0 ∧ real d ∈ {x/c<..x}. mangoldt d * χ_L (x
/ d))"
    using c by (intro sum_mono * mangoldt_nonneg) (auto intro!: chi_ge1
simp: field_simps)
  also have "... ≤ (∑ d | d > 0 ∧ real d ≤ x. mangoldt d * χ_L (x /
d))"
    by (intro sum_mono2) (auto intro!: mult_nonneg_nonneg mangoldt_nonneg
chi_nonneg)
  also have "... = ψ_L x"
    by (simp add: psi_L_def sum_upto_def)
  finally have "ψ x ≤ ψ (x / c) + ψ_L x"
    by - simp_all

have L: "3 * |real_of_int l| ≤ x" if "l ∈# L" for l
  using x2[OF that] x by linarith

have "ψ x ≤ ψ (x / c) + ψ_L x"
  by fact
also have "ψ_L x = α_L * x + psi_L_remainder x"

```

```

    using <x0 ≥ 3> x by (subst psi_L_eq) auto
  also have "|psi_L_remainder x| ≤ ln x * size L / 2 - C"
    using psi_L_remainder_bounds[of x] <x0 ≥ 3> x L by (simp add: C_def)
  hence "psi_L_remainder x ≤ ln x * size L / 2 - C"
    by linarith
  finally show "ψ x ≤ ψ (x / c) + αL * x + ln x * size L / 2 - C"
    by (simp add: algebra_simps)
qed

define m where "m = nat [log c (x / x0)]"
have "x > 0"
  using x x1 by simp

have "ψ x ≤ ψ x0 + (∑ k<m. αL * x / c ^ k + ln (x / c ^ k) * size
L / 2 - C)"
proof -
  have "ψ x ≤ ψ (x / c ^ m) + (∑ k<m. αL * (x / c ^ k) + ln (x / c
^ k) * size L / 2 - C)"
  proof (rule primes_psi_lower_rec)
    fix x :: real assume "x ≥ x0"
    thus "ψ x ≤ ψ (x / c) + (αL * x + ln x * size L / 2 - C)"
      using rec[of x] by (simp add: algebra_simps)
  next
    have "c ^ m = c powr real m"
      using c by (simp add: powr_realpow)
    also have "... ≤ c powr (log c (x / x0) + 1)"
      using c x <x0 ≥ 3> by (intro powr_mono) (auto simp: m_def)
    also have "... = c * x / x0"
      using c x <x0 ≥ 3> by (auto simp: powr_add)
    finally show "x0 * c ^ m ≤ x * c"
      using <x0 ≥ 3> by (simp add: field_simps)
  qed (use x1 c in auto)
  also have "ψ (x / c ^ m) ≤ ψ x0"
  proof (rule ψ_mono)
    have "x / x0 = c powr log c (x / x0)"
      using c x <x0 ≥ 3> by simp
    also have "... ≤ c powr m"
      unfolding m_def using c <x0 ≥ 3> x by (intro powr_mono) auto
    also have "... = c ^ m"
      using c by (simp add: powr_realpow)
    finally show "x / c ^ m ≤ x0"
      using <x0 ≥ 3> c by (simp add: field_simps)
  qed
  finally show ?thesis
    by simp
qed
also have "... = ψ x0 + (∑ k<m. αL * x / c ^ k + (ln x - k * ln c)
* size L / 2 - C)"
  using x(1) <x0 ≥ 3> c by (simp add: ln_div ln_realpow)

```



```

also have "... =  $\psi x_0 + \alpha_L * x * (\sum_{k < m} 1 / c ^ k) + \ln x * m * \text{size } L / 2 - \text{real } (\sum_{k < m} k) * \ln c * \text{size } L / 2 - C * m$ "
  by (simp add: sum_diff_distrib sum_subtractf sum.distrib sum_distrib_left
sum_distrib_right algebra_simps diff_divide_distrib sum_divide_distrib)
also have " $(\sum_{k < m} 1 / c ^ k) = (1 - (1 / c) ^ m) / (1 - 1 / c)$ "
  using sum_gp_strict[of "1/c" m] c by (simp add: field_simps)
also have "...  $\leq 1 / (1 - 1 / c)$ "
  using c by (intro divide_right_mono) auto
also have " $1 / (1 - 1/c) = c / (c - 1)$ "
  using c by (simp add: field_simps)
also have " $(\sum_{k < m} k) = \text{real } m * (\text{real } m - 1) / 2$ "
  by (induction m) (auto simp: field_simps)
finally have " $\psi x \leq \psi x_0 + c / (c - 1) * \alpha_L * x + \ln x * m * \text{size } L / 2 - \text{real } m * (\text{real } m - 1) / 2 * \ln c * \text{size } L / 2 - C * m$ "
  using < $\alpha_L \geq 0$ > < $x > 0$ > x1 by (simp add: mult_left_mono mult_right_mono mult_ac)
also have "... =  $\psi x_0 + c / (c - 1) * \alpha_L * x + m/2 * (\text{size } L * (\ln x - (\text{real } m - 1)/2 * \ln c + 2) - (\sum_{l \in \#L} \ln |l|))$ "
  by (simp add: algebra_simps C_def)
also have " $m/2 * (\text{size } L * (\ln x - (\text{real } m - 1)/2 * \ln c + 2) - (\sum_{l \in \#L} \ln |l|)) \leq m/2 * (\text{size } L * (3/2 * \ln x) - 0)$ "
proof (intro mult_left_mono diff_mono)
  have "real m  $\geq \log c (x / x_0)$ "
    using c < $x_0 \geq 3$ > x unfolding m_def by auto
  hence " $\ln x - (\text{real } m - 1)/2 * \ln c + 2 \leq \ln x - (\log c (x / x_0) - 1)/2 * \ln c + 2$ "
    using c by (intro diff_mono add_mono mult_right_mono divide_right_mono)
auto
also have "... =  $(\ln x + \ln x_0 + (\ln c + 4)) / 2$ "
  using c x < $x_0 \geq 3$ > by (simp add: log_def ln_div field_simps)
also have " $\ln x_0 \leq \ln x$ "
  using x x1 by simp
also have " $\ln c + 4 \leq \ln x$ "
proof -
  have "exp (4 :: real)  $\leq 55$ "
    by (approximation 10)
  hence "exp 4 * c  $\leq 55 * c$ "
    using c by (intro mult_right_mono) auto
  also have "55 * c  $\leq x_0$ "
    by fact
  also have "...  $\leq x$ "
    by fact
  finally have "exp (ln c + 4)  $\leq \exp (\ln x)$ "
    unfolding exp_add using c x1 x by (simp add: mult_ac)
  thus ?thesis
    by (simp only: exp_le_cancel_iff)

```

```

qed
also have "(ln x + ln x + ln x) / 2 = 3 / 2 * ln x"
  by simp
finally show "ln x - (real m - 1) / 2 * ln c + 2 ≤ 3 / 2 * ln x"
  by - simp
qed (auto intro!: sum_mset_nonneg simp: L_nonzero' Ints_nonzero_abs_ge1)
also have "m / 2 * (size L * (3/2 * ln x) - 0) = 3 / 4 * m * size L
* ln x"
  by simp
also have "... ≤ 3 / 4 * (ln x / ln c) * size L * ln x"
proof (intro mult_left_mono mult_right_mono)
  have "real m ≤ log c (x / x0) + 1"
    unfolding m_def using c x <x0 ≥ 3> by auto
  also have "... / 2 = (ln x / ln c + (1 - log c x0)) / 2"
    using <x0 ≥ 3> <x ≥ x0> c
    by (simp add: log_def ln_div field_simps)
  also have "1 - log c x0 ≤ 0"
    using x1 c by simp
  finally show "real m ≤ ln x / ln c" by - simp_all
qed (use x x1 in auto)
also have "... = (3 * size L) / (4 * ln c) * ln x ^ 2"
  by (simp add: power2_eq_square)
finally show "ψ x ≤ c / (c - 1) * αL * x + (3 * size L) / (4 * ln c)
* ln x ^ 2 + ψ x0"
  by (simp add: algebra_simps)
qed
end

```

1.5 Final results

```

theorem psi_lower_ge_9:
  assumes x: "x ≥ 41"
  shows "ψ x ≥ 0.9 * x"
proof (cases "x ≥ 900")
  case False
  have "∀x∈{real 41..real 900}. primes_psi x ≥ 0.9 * x"
    by (rule check_psi_lower_correct[where prec = 16]) eval
  from bspec[OF this, of x] show ?thesis
    using assms False by simp
next
  case x: True
  define f :: "real ⇒ real"
    where "f = (λx. 0.02128 * x - 2.5 * ln x - 1.6)"
  have "0 ≤ f 900"
    unfolding f_def by (approximation 10)
  also have "f 900 ≤ f x"
    using x
  proof (rule DERIV_nonneg_imp_nondecreasing, goal_cases)

```

```

    case (1 t)
    have "(f has_real_derivative (0.02128 - 2.5 / t)) (at t)"
      unfolding f_def using 1 by (auto intro!: derivative_eq_intros)
    moreover have "0.02128 - 2.5 / t ≥ 0"
      using 1 by (auto simp: field_simps)
    ultimately show ?case
      by blast
  qed
  finally have "0.9 * x ≤ 0.9 * x + f x"
    by linarith
  also have "... = 0.92128 * x - 2.5 * ln x - 1.6"
    by (simp add: f_def)
  also have "... ≤ ψ x"
    by (rule psi_lower_bound_precise) (use x in auto)
  finally show ?thesis .
qed

theorem primes_theta_ge_82:
  assumes "x ≥ 97"
  shows "∃ x ≥ 0.82 * x"
proof (cases "x ≥ 46000")
  case False
  have "∀ x ∈ {real 97..real 46000}. ∃ x ≥ 0.82 * x"
    by (rule check_theta_lower_correct[where prec = 20]) eval
  from bspec[OF this, of x] show ?thesis
    using False assms by simp
next
  case True
  with assms have x: "x ≥ 46000"
    by auto
  define f :: "real ⇒ real"
    where "f = (λx. 0.10128 * x - 2.5 * ln x - 2 * ln x * sqrt x - 1.6)"
  have "0 ≤ f 46000"
    unfolding f_def by (approximation 30)
  also have "f 46000 ≤ f x"
    using x
  proof (rule DERIV_nonneg_imp_nondecreasing, goal_cases)
    case (1 t)
    define D where "D = 0.10128 - 2.5 / t - 2 * sqrt t / t - ln t / sqrt
t"
    have deriv: "(f has_real_derivative D) (at t)"
      unfolding f_def
      by (rule derivative_eq_intros refl | use 1 in force)+
      (simp add: field_simps D_def)
    have "0.10128 - D = 2.5 / t + 2 / sqrt t + ln t / sqrt t"
      using 1 by (simp add: D_def field_simps del: div_add div_diff div_mult_self1
div_mult_self2 div_mult_self3 div_mult_self4)
    also have "... ≤ 2.5 / 46000 + 2 / 214 + ln t / sqrt t"
      using 1 by (intro add_mono) (auto simp: real_le_sqrt)
  end
end

```

```

also have "ln t / sqrt t ≤ ln 46000 / sqrt 46000"
  using 1(1)
proof (rule DERIV_nonpos_imp_nonincreasing, goal_cases)
  case (1 u)
  have "((λt. ln t / sqrt t) has_real_derivative ((2 - ln u) / (2
* u * sqrt u))) (at u)"
    by (rule derivative_eq_intros refl | use 1 in force)+
      (use 1 in <simp add: field_simps>)
  moreover {
    have "2 ≤ ln (10::real)"
      by (approximation 30)
    also have "... ≤ ln u"
      using 1 by simp
    finally have "ln u ≥ 2" .
  }
  hence "((2 - ln u) / (2 * u * sqrt u)) ≤ 0"
    using 1 by (intro divide_nonpos_nonneg) auto
  ultimately show ?case
    by blast
qed
also have "... ≤ 0.0501"
  by (approximation 30)
also have "2.5 / 46000 + 2 / 214 + 0.0501 ≤ (0.10128 :: real)"
  by simp
finally have "D ≥ 0"
  by simp
with deriv show ?case by blast
qed

finally have "0.82 * x ≤ 0.82 * x + f x"
  by linarith
also have "... = 0.92128 * x - 2.5 * ln x - 2 * ln x * sqrt x - 1.6"
  by (simp add: f_def)
also have "... ≤ 0.92128 * x - 2.5 * ln x - 1.6 + ϑ x - ψ x"
  using ψ_minus_ϑ_bound[of x] x by simp
also have "0.92128 * x - 2.5 * ln x - 1.6 ≤ ψ x"
  by (rule psi_lower_bound_precise) (use x in auto)
finally show ?thesis by simp
qed

corollary primorial_ge_exp_82:
  assumes "x ≥ 97"
  shows "primorial x ≥ exp (0.82 * x)"
proof -
  have "primorial x = exp (ϑ x)"
    using ln_primorial[of x] primorial_pos[of x]
    by (metis exp_ln of_nat_0_less_iff)
  also have "... ≥ exp (0.82 * x)"

```

```

    using primes_theta_ge_82[OF assms] by simp
  finally show ?thesis .
qed

theorem primes_psi_le_111:
  assumes "x ≥ 0"
  shows "ψ x ≤ 1.11 * x"
proof -
  have "∀x∈{real 0..real 146000}. primes_psi x ≤ 1.04 * x"
  proof (rule check_psi_upper_correct[where prec = 16])
    show "check_psi_upper (104 / 102) 16 0 146000"
      by eval
  qed auto
  hence initial: "primes_psi x ≤ 1.04 * x" if "x ∈ {0..146000}" for x
    using that by auto

  show ?thesis
  proof (cases "x ≥ 146000")
    case False
    thus ?thesis
      using initial[of x] assms by simp
  next
    case x: True
    define L :: "int multiset" where "L = {#1, -2, -3, -5, 30#}"
    have [simp]: "set_mset L = {1, -2, -3, -5, 30}" "size L = 5"
      by (simp_all add: L_def)
    interpret balanced_chebyshev_multiset L
      by unfold_locales (auto simp: L_def)
    define x0 :: real where "x0 = Max ({3, 55 * 6} ∪ {3 * |real_of_int 1| | 1. 1 ∈# L})"

    have x0: "x0 = 330"
    proof -
      have "x0 = Max ({3, 55 * 6} ∪ {3 * |real_of_int 1| | 1. 1 ∈# L})"
        unfolding x0_def ..
      also have "{3 * |real_of_int 1| | 1. 1 ∈# L} = (λ1. 3 * |of_int 1|)"
        ' set_mset L"
        by blast
      finally show ?thesis
        by simp
    qed

    define f :: "real ⇒ real"
      where "f = (λt. 2.093 * ln t ^ 2 + 343.2 - 0.0044 * t)"

    have alpha_L: "alpha_L = ln 2 / 2 - (ln 30 / 30 - ln 5 / 5 - ln 3 / 3)"
      unfolding alpha_L_def by (simp add: L_def)

```

```

have "alpha_L ≥ 0"
  unfolding alpha_L by (approximation 10)
have period: "period = 30"
  by (simp add: period_def)

have "ψ x ≤ 6 / (6 - 1) * alpha_L * x + (3 * size L) / (4 * ln 6)
* ln x ^ 2 + ψ x0"
  unfolding x0_def
proof (rule psi_upper_bound; (unfold period)?)
  show "chi_L (real n) ≥ 0" if "n ∈ {0..<30}" for n
    unfolding chi_L_def
    using that unfolding atLeastLessThan_nat_numeral pred_numeral_simps
arith_simps
    by (auto simp: L_def)
next
  show "chi_L (real n) ≥ 1" if "real n ∈ {1..<6}" for n
  proof -
    have "n ∈ {1..<6}"
      using that by auto
    also have "{1..<6} = {1,2,3,4,5::nat}"
      by auto
    finally show ?thesis
      unfolding chi_L_def by (elim insertE) (auto simp: L_def)
  qed
qed (use <alpha_L ≥ 0> x in auto)

also have "... = 6/5 * alpha_L * x + 15 / (4 * ln 6) * (ln x)^2 + ψ
x0"
  by simp
also have "ψ x0 ≤ 343.2"
  using initial[of x0] by (simp add: x0)
also have "6/5 * alpha_L ≤ 1.1056"
  unfolding alpha_L by (approximation 30)
also have "15 / (4 * ln 6 :: real) ≤ 2.093"
  by (approximation 20)
finally have "ψ x ≤ 1.1056 * x + 2.093 * ln x ^ 2 + 343.2"
  using x by - simp_all
also have "... = 1.11 * x + f x"
  by (simp add: algebra_simps f_def)
also have "f x ≤ f 146000"
  using x
proof (rule DERIV_nonpos_imp_nonincreasing, goal_cases)
  case (1 t)
  define f' :: "real ⇒ real"
    where "f' = (λt. 4.186 * ln t / t - 0.0044)"
  have "(f has_field_derivative f' t) (at t)"
    using 1 unfolding f_def f'_def
    by (auto intro!: derivative_eq_intros)
  moreover {

```

```

    have "f' t ≤ f' 146000"
      using 1(1)
    proof (rule DERIV_nonpos_imp_nonincreasing, goal_cases)
      case (1 u)
        have "(f' has_field_derivative (4.186 * (1 - ln u) / u ^ 2))
          (at u)"
          using 1 unfolding f'_def
          by (auto intro!: derivative_eq_intros simp: field_simps power2_eq_square)
        moreover have "4.186 * (1 - ln u) / u ^ 2 ≤ 0"
          using 1 exp_le by (auto intro!: divide_nonpos_nonneg simp:
ln_ge_iff)
        ultimately show ?case
          by blast
      qed
      also have "... ≤ 0"
        unfolding f'_def by (approximation 10)
      finally have "f' t ≤ 0" .
    }
    ultimately show ?case
      by blast
  qed
  also have "f 146000 ≤ 0"
    unfolding f_def by (approximation 10)
  finally show ?thesis
    by - simp_all
qed
qed

```

```

corollary primes_theta_le_111:
  assumes "x ≥ 0"
  shows "∃ x ≤ 1.11 * x"
  using primes_psi_le_111[OF assms] ∅_le_ψ[of x]
  by linarith

```

As an easy corollary, we obtain Bertrand's postulate: For any real number $x > 1$, the interval $(x, 2x)$ contains at least one prime.

```

corollary bertrands_postulate:
  assumes "x > 1"
  shows "∃ p. prime p ∧ real p ∈ {x <.. $2x$ }"
proof (cases "x ≥ 7")
  case False
  consider "x ∈ {1 <.. $2$ }" | "x ∈ {2 <.. $3$ }" | "x ∈ {3 <.. $5$ }" | "x ∈ {5 <.. $7$ }"
  using False assms by force
  thus ?thesis
  proof cases
    case 1
    thus ?thesis by (intro exI[of _ 2]; simp)
  next
    case 2

```

```

    thus ?thesis by (intro exI[of _ 3]; simp)
next
  case 3
  thus ?thesis by (intro exI[of _ 5]; simp)
next
  case 4
  thus ?thesis by (intro exI[of _ 7]; simp)
qed
next
case x: True
have fin: "finite {p. prime p ∧ real p ≤ 1.999 * x}"
  by (rule finite_subset[of _ "{..nat ⌊2*x⌋}"])
  (use x in <auto simp: le_nat_iff le_floor_iff>)

have "∅ (1.999 * x) > 1.11 * x"
proof (cases "x ≥ 49")
  case False
  have "∀x∈{real 11..real 100}. ∅ x ≥ 0.556 * x"
    by (rule check_theta_lower_correct[where prec = 10]) eval
  from bspec[OF this, of "1.999*x"] show ?thesis
    using False x by simp
next
  case True
  thus ?thesis
    using primes_theta_ge_82[of "1.999*x"] True by auto
qed

have "∅ x ≤ 1.11 * x"
  by (rule primes_theta_le_111) (use x in auto)
also have "... < ∅ (1.999 * x)"
  by fact
finally have "∅ (1.999 * x) > ∅ x" .

have "{p. prime p ∧ real p ∈ {x<..1.999*x}} ≠ {}"
proof
  assume eq: "{p. prime p ∧ real p ∈ {x<..1.999*x}} = {}"
  have "∅ (1.999 * x) = ∅ x + (∑ p | prime p ∧ real p ∈ {x<..1.999*x}.
ln p)"
    unfolding primes_theta_def prime_sum_upto_def
    by (rule sum.union_disjoint' [symmetric]) (use fin in auto)
  also note eq
  finally show False
    using <∅ (1.999 * x) > ∅ x> by simp
qed
thus ?thesis
  by auto
qed

unbundle no_prime_counting_notation

```


end