

# The Perfect Number Theorem

Mark IJbema

December 14, 2021

## Abstract

This document presents the formal proof of the Perfect Number Theorem. The result can also be found as number 70 on the list of “top 100 mathematical theorems” [Wie]. This document was produced as result of a B.Sc. Thesis under supervision of Jaap Top and Wim H. Hesselink (University of Groningen) in 2009.

## Contents

<b>1</b>	<b>Basics needed</b>	<b>1</b>
<b>2</b>	<b>Sum of divisors function</b>	<b>2</b>
<b>3</b>	<b>Perfect Number Theorem</b>	<b>6</b>

## 1 Basics needed

**theory** *PerfectBasics*

**imports** *Main HOL-Computational-Algebra.Primes HOL-Algebra.Exponent*  
**begin**

**lemma** *exp-is-max-div:*

**assumes**  $m0: m \neq 0$  **and**  $p: \text{prime } p$   
**shows**  $\sim p \text{ dvd } (m \text{ div } (p^{\wedge}(\text{multiplicity } p \ m))))$

**proof** (*rule ccontr*)

**assume**  $\sim \sim p \text{ dvd } (m \text{ div } (p^{\wedge}(\text{multiplicity } p \ m))))$

**hence**  $a: p \text{ dvd } (m \text{ div } (p^{\wedge}(\text{multiplicity } p \ m))))$  **by** *auto*

**from**  $m0$  **have**  $p^{\wedge}(\text{multiplicity } p \ m) \text{ dvd } m$  **by** (*auto simp add: multiplicity-dvd*)

**with**  $a$  **have**  $p^{\wedge}(\text{Suc } (\text{multiplicity } p \ m)) \text{ dvd } m$

**by** (*subst (asm) dvd-div-iff-mult*) *auto*

**with**  $m0$  **show** *False*

**by** (*subst (asm) power-dvd-iff-le-multiplicity*) *auto*

**qed**

**lemma** *coprime-multiplicity:*

**assumes** *prime* ( $p::\text{nat}$ ) **and**  $m > 0$

**shows**  $\text{coprime } p \ (m \ \text{div} \ (p \wedge \text{multiplicity } p \ m))$   
**proof** (*rule ccontr*)  
**assume**  $\neg \text{coprime } p \ (m \ \text{div} \ p \wedge \text{multiplicity } p \ m)$   
**with**  $\langle \text{prime } p \rangle$  **have**  $\exists q. \text{prime } q \wedge q \ \text{dvd} \ p \wedge q \ \text{dvd} \ m \ \text{div} \ p \wedge \text{multiplicity } p \ m$   
**by** (*metis dvd-refl prime-imp-coprime*)  
**with**  $\langle \text{prime } p \rangle$  **have**  $\exists q. q = p \wedge q \ \text{dvd} \ m \ \text{div} \ p \wedge \text{multiplicity } p \ m$   
**by** (*metis not-prime-1 prime-nat-iff*)  
**then have**  $p \ \text{dvd} \ m \ \text{div} \ p \wedge \text{multiplicity } p \ m$   
**by** *auto*  
**with** *assms* **show** *False*  
**by** (*auto simp add: exp-is-max-div*)  
**qed**

**theorem** *simplify-sum-of-powers*:  $(x - 1 :: \text{nat}) * (\sum_{i=0}^n n \cdot x^i) = x^{n+1} - 1$  (*is ?l = ?r*)  
**proof** (*cases*)  
**assume**  $n = 0$   
**thus**  $?l = x^{n+1} - 1$  **by** *auto*  
**next**  
**assume**  $n \neq 0$   
**hence**  $n0: n > 0$  **by** *auto*  
**have**  $?l = (x :: \text{nat}) * (\sum_{i=0}^n n \cdot x^i) - (\sum_{i=0}^n n \cdot x^i)$   
**by** (*metis diff-mult-distrib nat-mult-1*)  
**also have**  $\dots = (\sum_{i=0}^n n \cdot x^{(\text{Suc } i)}) - (\sum_{i=0}^n n \cdot x^i)$   
**by** (*simp add: sum-distrib-left*)  
**also have**  $\dots = (\sum_{i=\text{Suc } 0}^n \text{Suc } n \cdot x^i) - (\sum_{i=0}^n n \cdot x^i)$   
**by** (*metis sum.shift-bounds-cl-Suc-ivl*)  
**also have**  $\dots = ((\sum_{i=\text{Suc } 0}^n n \cdot x^i) + x^{(\text{Suc } n)}) - (x^0 + (\sum_{i=\text{Suc } 0}^n n \cdot x^i))$   
**by** (*simp add: sum.union-disjoint diff-add-inverse sum.atLeast-Suc-atMost*)  
**finally show**  $?thesis$  **by** *auto*  
**qed**  
**end**

## 2 Sum of divisors function

**theory** *Sigma*  
**imports** *PerfectBasics HOL-Library.Infinite-Set*  
**begin**

**definition** *divisors* ::  $\text{nat} \Rightarrow \text{nat set}$  **where**  
 $\text{divisors } m \equiv \{n \cdot n \ \text{dvd} \ m\}$

**abbreviation** *sigma* ::  $\text{nat} \Rightarrow \text{nat}$  **where**  
 $\text{sigma } m \equiv \sum (\text{divisors}(m))$

**lemma** *divisors-eq-dvd[iff]*:  $(a \in \text{divisors}(n)) \iff (a \ \text{dvd} \ n)$   
**by** (*simp add: divisors-def*)

**lemma** *finite-divisors* [*simp*]:  
**assumes**  $n > 0$  **shows** *finite* (*divisors*  $n$ )  
**by** (*simp add: assms divisors-def*)

**lemma** *divs-of-zero-UNIV* [*simp*]: *divisors*( $0$ ) = *UNIV*  
**by**(*auto simp add: divisors-def*)

**lemma** *sigma0* [*simp*]: *sigma*( $0$ ) =  $0$   
**by** *simp*

**lemma** *sigma1* [*simp*]: *sigma*(*Suc*  $0$ ) =  $1$   
**by** (*simp add: sum-eq-Suc0-iff*)

**lemma** *prime-divisors*: *prime*  $p \iff$  *divisors*  $p = \{1, p\} \wedge p > 1$   
**by** (*auto simp add: divisors-def prime-nat-iff*)

**lemma** *prime-imp-sigma*: *prime* ( $p::nat$ )  $\implies$  *sigma*( $p$ ) =  $p+1$   
**proof** –  
**assume** *prime* ( $p::nat$ )  
**hence**  $p > 1 \wedge$  *divisors*( $p$ ) =  $\{1, p\}$  **by** (*simp add: prime-divisors*)  
**hence**  $p > 1 \wedge$  *sigma*( $p$ ) =  $\sum \{1, p\}$  **by** (*auto simp only: divisors-def*)  
**thus** *sigma*( $p$ ) =  $p+1$  **by** *simp*  
**qed**

**lemma** *sigma-third-divisor*:  
**assumes**  $1 < a$   $a < n$   $a \text{ dvd } n$   
**shows**  $1+a+n \leq$  *sigma*( $n$ )  
**proof** –  
**from** *assms* **have**  $\{1, a, n\} \leq$  *divisors*  $n$   
**by** *auto*  
**hence**  $\sum \{1, a, n\} \leq$  *sigma*  $n$   
**by** (*meson*  $\langle a < n \rangle$  *finite-divisors order.strict-trans1 sum-mono2 zero-le*)  
**with** *assms* **show** *thesis* **by** *auto*  
**qed**

**proposition** *prime-iff-sigma*: *prime*  $n \iff$  *sigma*( $n$ ) = *Suc*  $n$   
**proof**  
**assume**  $L$ : *sigma*( $n$ ) = *Suc*  $n$   
**then** **have**  $n > 1$   
**using** *less-linear sigma1* **by** *fastforce*  
**moreover**  
**have**  $m = \text{Suc } 0$  **if**  $m \text{ dvd } n$   $m \neq n$  **for**  $m$   
**proof** –  
**have**  $0 < m$   
**using** *that* **by** *auto*  
**then** **have**  $\neg 1 + m + n \leq 1 + n$   
**by** *linarith*  
**then** **show** *thesis*

**using** *sigma-third-divisor* [of *m*]  
**by** (*metis L One-nat-def Suc-lessD Suc-lessI*  $\langle n > 1 \rangle$  *dvd-imp-le*  $\langle 0 < m \rangle$   
*less-le plus-1-eq-Suc that*)  
**qed**  
**then have** *divisors*  $n = \{n, 1\}$   
**by** (*auto simp: divisors-def*)  
**ultimately show** *prime*  $n$   
**by** (*simp add: insert-commute prime-divisors*)  
**qed** (*use prime-divisors in auto*)

**lemma** *dvd-prime-power-iff*:  
**fixes**  $p::nat$   
**assumes** *prime*: *prime*  $p$   
**shows**  $\{d. d \text{ dvd } p^n\} = (\lambda k. p^k) \text{ ' } \{0..n\}$   
**using** *divides-primelow-nat prime* **by** (*auto simp add: le-imp-power-dvd*)

**lemma** *rewrite-sum-of-powers*:  
**assumes**  $p: (p::nat) > 1$   
**shows**  $\sum ((\wedge) p \text{ ' } \{0..n\}) = (\sum i = 0 .. n . p^i)$  (**is**  $?l = ?r$ )  
**by** (*metis inj-on-def p power-inject-exp sum.reindex-cong*)

**lemma** *sum-of-powers-int*:  $(x - 1::int) * (\sum i=0..n . x^i) = x^{Suc\ n} - 1$   
**by** (*metis atLeast0AtMost lessThan-Suc-atMost power-diff-1-eq*)

**lemma** *sum-of-powers-nat*:  $(x - 1::nat) * (\sum i=0..n . x^i) = x^{Suc\ n} - 1$   
**(is**  $?l = ?r$ )  
**proof** (*cases*  $x = 0$ )  
**case** *False*  
**then have**  $int\ ((x - 1) * \text{sum } ((\wedge) x) \{0..n\}) = int\ (x^{Suc\ n} - 1)$   
**using** *sum-of-powers-int* [of  $int\ x\ n$ ] **by** (*simp add: of-nat-diff*)  
**then show**  $?thesis$   
**using** *of-nat-eq-iff* **by** *blast*  
**qed** *auto*

**theorem** *sigma-primelow*:  
**assumes** *prime*  $p$   
**shows**  $(p - 1) * \text{sigma}(p^e) = p^{e+1} - 1$   
**proof** –  
**have**  $\text{sigma}(p^e) = (\sum i=0..e . p^i)$   
**using** *assms divisors-def dvd-prime-power-iff prime-nat-iff rewrite-sum-of-powers*  
**by** *auto*  
**thus**  $(p - 1) * \text{sigma}(p^e) = p^{e+1} - 1$   
**using** *sum-of-powers-nat* **by** *auto*  
**qed**

**proposition** *sigma-primelow-two*:  $\text{sigma}(2^n) = 2^{n+1} - 1$   
**proof** –  
**have**  $(2 - 1) * \text{sigma}(2^n) = 2^{n+1} - 1$

by (auto simp only: sigma-primpower two-is-prime-nat)  
 thus ?thesis by simp  
 qed

**lemma** prodsums-eq-sumprods:

fixes  $p :: \text{nat}$  and  $m :: \text{nat}$

assumes coprime  $p$   $m$

shows  $\sum ((\lambda k. p^k) \text{ ` } \{0..n\}) * \text{sigma } m = \sum \{p^k * b \mid b. k \leq n \wedge b \text{ dvd } m\}$   
 (is ?lhs = ?rhs)

**proof** –

have coprime  $p$   $x$  if  $x \text{ dvd } m$  for  $x$

using assms by (rule coprime-imp-coprime) (auto intro: dvd-trans that)

then have coprime  $(p^f)$   $x$  if  $x \text{ dvd } m$  for  $x$   $f$

using that by simp

then have ?lhs =  $\sum \{a * b \mid a \text{ dvd } m. (\exists f. a = p^f \wedge f \leq n) \wedge b \text{ dvd } m\}$

apply (subst sum-mult-sum-if-inj [OF mult-inj-if-coprime-nat])

apply (force intro!: sum.cong)+

done

also have ... = ?rhs

by (blast intro: sum.cong)

finally show ?thesis .

qed

**lemma** div-decomp-comp:

fixes  $a :: \text{nat}$

shows coprime  $m$   $n \implies a \text{ dvd } m * n \iff (\exists b \text{ c. } a = b * c \wedge b \text{ dvd } m \wedge c \text{ dvd } n)$

by (auto simp only: division-decomp mult-dvd-mono)

**theorem** sigma-semimultiplicative:

assumes  $p$ : prime  $p$  and  $cop$ : coprime  $p$   $m$

shows  $\text{sigma } (p^n) * \text{sigma } m = \text{sigma } (p^n * m)$  (is ?lhs = ?rhs)

**proof** –

from  $cop$  have  $cop2$ : coprime  $(p^n)$   $m$

by simp

from  $p$  have ?lhs =  $\sum ((\lambda f. p^f) \text{ ` } \{0..n\}) * \text{sigma } m$

using divisors-def dvd-prime-power-iff by auto

also from  $cop$  have ... =  $(\sum \{p^f * b \mid f \leq n \wedge b \text{ dvd } m\})$

by (auto simp add: prodsums-eq-sumprods prime-nat-iff)

also have ... =  $(\sum \{a * b \mid a \text{ dvd } (p^n) \wedge b \text{ dvd } m\})$

by (metis (no-types, opaque-lifting) le-imp-power-dvd divides-primpower-nat  $p$ )

also have ... =  $\sum \{c. c \text{ dvd } (p^n * m)\}$

using  $cop2$  div-decomp-comp by auto

finally show ?thesis

by (simp add: divisors-def)

qed

end

### 3 Perfect Number Theorem

```
theory Perfect
imports Sigma
begin
```

```
definition perfect :: nat => bool where
  perfect m  $\equiv$  m > 0  $\wedge$  2 * m = sigma m
```

```
theorem perfect-number-theorem:
```

```
  assumes even: even m and perfect: perfect m
```

```
  shows  $\exists$  n . m = 2n * (2(n+1) - 1)  $\wedge$  prime ((2::nat)(n+1) - 1)
```

```
proof
```

```
  from perfect have m0: m > 0 by (auto simp add: perfect-def)
```

```
  let ?n = multiplicity 2 m
  let ?A = m div 2?n
  let ?np = (2::nat)(?n+1) - 1
```

```
  from even m0 have n1: ?n >= 1 by (simp add: multiplicity-ge1)
```

```
  have 2?n dvd m by (rule multiplicity-dvd)
```

```
  hence m = 2?n * ?A by (simp only: dvd-mult-div-cancel)
```

```
  with m0 have mdef: m = 2?n * ?A  $\wedge$  coprime 2 ?A
```

```
    using multiplicity-decompose [of m 2] by simp
```

```
  moreover with m0 have a0: ?A > 0 by (metis nat-0-less-mult-iff)
```

```
  moreover
```

```
  { from perfect have 2 * m = sigma(m) by (simp add: perfect-def)
```

```
    with mdef have 2(?n+1) * ?A = sigma(2?n * ?A) by auto
```

```
  } ultimately have 2(?n+1) * ?A = sigma(2?n) * sigma(?A)
```

```
    by (simp add: sigma-semimultiplicative)
```

```
  hence formula: 2(?n+1) * ?A = (?np) * sigma(?A)
```

```
    by (simp only: sigma-prime-power-two)
```

```
  from n1 have (2::nat)(?n+1) >= 22 by (simp only: power-increasing)
```

```
  hence nplarger: ?np >= 3 by auto
```

```
  let ?B = ?A div ?np
```

```
  from formula have ?np dvd ?A * 2(?n+1)
```

```
    by (auto simp add: ac-simps)
```

```
  then have ?np dvd ?A
```

```
    using coprime-diff-one-left-nat [of 2(multiplicity 2 m + 1)]
```

```
    by (auto simp add: coprime-dvd-mult-left-iff)
```

```
  then have bdef: ?np * ?B = ?A
```

```
    by simp
```

```
  with a0 have b0: ?B > 0 by (metis gr0I mult-is-0)
```

```
  from nplarger a0 have bsmallera: ?B < ?A by auto
```

```

have ?B = 1
proof (rule ccontr)
  assume ?B ≠ 1
  with b0 bsmallera have 1 < ?B ?B < ?A by auto
  moreover from bdef have ?B : divisors ?A
    by (metis divisors-eq-dvd dvd-triv-right)
  ultimately have 1 + ?B + ?A ≤ sigma ?A
    using sigma-third-divisor by blast
  with nplarger have ?np*(1 + ?A + ?B) ≤ ?np*(sigma ?A)
    by (auto simp only: nat-mult-le-cancel1)
  with bdef have ?np + ?A * ?np + ?A * 1 ≤ ?np*(sigma ?A)
    by (simp add: mult.commute distrib-left)
  hence ?np + ?A*(?np + 1) ≤ ?np*(sigma ?A) by (simp only: add-mult-distrib2)
  with nplarger have 2^(?n+1)*?A < ?np*(sigma ?A) by (simp add: mult.commute)
  with formula show False by auto
qed

with bdef have adef: ?A = ?np by auto
with formula have ?np*2^(?n+1) = ?np * sigma(?A) by auto
with nplarger adef have ?A + 1 = sigma(?A) by auto
with a0 have prime ?A
  by (simp add: prime-iff-sigma)
with mdef adef show m = 2^?n * ?np ∧ prime ?np by simp
qed

theorem Euclid-book9-prop36:
  assumes p: prime (2^(n+1) - (1::nat))
  shows perfect (2^n * (2^(n+1) - 1))
  unfolding perfect-def
proof (intro conjI; simp)
  from assms show 2 * 2^n > Suc 0 by (auto simp add: prime-nat-iff)
next
  have 2 ≠ ((2::nat)^(n+1) - 1) by simp arith
  then have coprime (2::nat) (2^(n+1) - 1)
    by (metis p primes-coprime-nat two-is-prime-nat)
  moreover with p have 2^(n+1) - 1 > (0::nat)
    by (auto simp add: prime-nat-iff)
  ultimately have sigma (2^n*(2^(n+1) - 1)) = (sigma(2^n))*(sigma(2^(n+1)
- 1))
    by (metis sigma-semimultiplicative two-is-prime-nat)
  also from assms have ... = (sigma(2^n))*(2^(n+1))
    by (auto simp add: prime-imp-sigma)
  also have ... = (2^(n+1) - 1)*(2^(n+1)) by (simp add: sigma-prime-power-two)
  finally show 2*(2^n * (2*2^n - Suc 0)) = sigma(2^n*(2*2^n - Suc 0)) by
auto
qed

end

```

## References

- [Wie] Freek Wiedijk. Formalizing 100 theorems. <http://www.cs.ru.nl/~freek/100/>.