The Perfect Number Theorem

Mark IJbema

August 16, 2018

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

1 Basics needed

theory *PerfectBasics*
imports *Main HOL−Computational-Algebra.Primes HOL−Algebra.Exponent*
begin

lemma *sum-mono2-nat*: *finite* (*B*::*nat set*) $\Longrightarrow$ *A* <= *B* $\Longrightarrow$ $\sum$ *A* <= $\sum$ *B*
by (*auto simp add*: *sum-mono2*)

lemma *multiplicity-0* [*simp*]: *multiplicity 0 x* = *0*
by (*cases x* = *0*) (*auto intro*: *not-dvd-imp-multiplicity-0*)

lemma *exp-is-max-div*:
assumes *m0*:*m* ≠ *0* **and** *p*: *prime p*
shows ~ *p dvd* (*m div* (*p*^(*multiplicity p m*)))
proof (*rule ccontr*)
assume ~ ~ *p dvd* (*m div* (*p*^(*multiplicity p m*)))
hence *a*:*p dvd* (*m div* (*p*^(*multiplicity p m*))) **by** *auto*
from *m0* **have** *p*^(*multiplicity p m*) *dvd m* **by** (*auto simp add*: *multiplicity-dvd*)

with *a* **have** *p^Suc (multiplicity p m) dvd m*
by (*subst (asm) dvd-div-iff-mult*) *auto*
with *m0 p* **show** *False*
by (*subst (asm) power-dvd-iff-le-multiplicity*) *auto*
qed

lemma *coprime-multiplicity*:
assumes *prime (p::nat)* **and** *m > 0*
shows *coprime p (m div (p ^ multiplicity p m))*
proof (*rule ccontr*)
assume ¬ *coprime p (m div p ^ multiplicity p m)*
with ‹*prime p*› **have** ∃ *q*. *prime q* ∧ *q dvd p* ∧ *q dvd m div p ^ multiplicity p m*
by (*metis dvd-refl prime-imp-coprime*)
with ‹*prime p*› **have** ∃ *q*. *q = p* ∧ *q dvd m div p ^ multiplicity p m*
by (*metis not-prime-1 prime-nat-iff*)
then have *p dvd m div p ^ multiplicity p m*
by *auto*
with *assms* **show** *False*
by (*auto simp add*: *exp-is-max-div*)
qed

lemma *add-mult-distrib-three*: *(x::nat)*(a+b+c)=x*a+x*b+x*c*
proof −
have *(x::nat)*(a+b+c) = x*((a+b)+c)* **by** *auto*
hence *x*(a+b+c) = x*(a+b)+x*c* **by** (*simp add*: *algebra-simps*)
thus *x*(a+b+c) = x*a+x*b+x*c* **by** (*simp add*: *algebra-simps*)
qed

lemma *nat-interval-minus-zero*: *{0..Suc n} = {0} Un {Suc 0..Suc n}* **by** *auto*
lemma *nat-interval-minus-zero2*:
assumes *n>0*
shows *{0..n} = {0} Un {Suc 0..n}* **by** (*auto simp add*: *nat-interval-minus-zero*)

theorem *simplify-sum-of-powers*: *(x − 1::nat) * (∑ i=0 .. n . 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~=0*
hence *n0*: *n>0* **by** *auto*
have *?l = (x::nat)*(∑ i=0 .. n . x^i) − (∑ i=0 .. n . x^i)*
by (*metis diff-mult-distrib nat-mult-1*)
also have *... = (∑ i=0 .. n . x^(Suc i)) − (∑ i=0 .. n . x^i)*
by (*simp add*: *sum-distrib-left*)
also have *... = (∑ i=Suc 0 .. Suc n . x^i) − (∑ i=0 .. n . x^i)*
by (*metis sum-shift-bounds-cl-Suc-ivl*)
also with *n0*
have *... = ((∑ i=Suc 0 .. n. x^i)+x^(Suc n)) − (x^0 + (∑ i=Suc 0 .. n. x^i))*

by (*auto simp add*: *sum.union-disjoint nat-interval-minus-zero2*)
finally show *?thesis* **by** *auto*
qed

end

2 Sum of divisors function

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

definition *divisors* :: *nat* => *nat set* **where**
divisors (*m*::*nat*) == {*n* . *n dvd m*}

definition *sigma* :: *nat* => *nat* **where**
sigma m == $\sum$ *n* | *n dvd m* . *n*

lemma *sigma-divisors*: *sigma*(*n*) = $\sum$ (*divisors*(*n*))
by (*auto simp*: *sigma-def divisors-def*)

lemma *divisors-eq-dvd*[*iff*]: (*a*:*divisors*(*n*)) = (*a dvd n*)
by(*simp add*: *divisors-def*)

lemma *mult-divisors*: (*a*::*nat*)∗*b*=*c*==>*a*: *divisors c*
by (*unfold divisors-def dvd-def*,*blast*)
lemma *mult-divisors2*: (*a*::*nat*)∗*b*=*c*==>*b*: *divisors c*
by (*unfold divisors-def dvd-def*,*auto*)

lemma *divisorsfinite*[*simp*]:
assumes *n>0*
shows *finite* (*divisors n*)
proof −
from *assms* **have** *divisors n* = {*m* . *m dvd n* & *m* <= *n*}
by (*auto simp only*:*divisors-def dvd-imp-le*)
hence *divisors n* <= {*m* . *m*<=*n*} **by** *auto*
thus *finite* (*divisors n*)
by (*metis finite-Collect-le-nat finite-subset*)
qed

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

lemma *sigma0*[*simp*]: *sigma*(*0*) = *0*
by (*simp add*: *sigma-def*)
lemma *sigma1*[*simp*]: *sigma*(*1*) = *1*
by (*simp add*: *sigma-def*)

lemma *prime-divisors*: *prime* (*p*::*nat*) ⟷ *divisors p* = {*1*,*p*} & *p>1*

by (*auto simp add*: *divisors-def prime-nat-iff*)

lemma *prime-imp-sigma*: *prime* (*p*::*nat*) ==> *sigma*(*p*) = *p*+*1*
proof −
assume *prime* (*p*::*nat*)
hence *p*>*1* ∧ *divisors*(*p*) = {*1*,*p*} **by** (*simp add*: *prime-divisors*)
hence *p*>*1* ∧ *sigma*(*p*) = ∑ {*1*,*p*} **by** (*auto simp only*: *sigma-divisors divisors-def*)
thus *sigma*(*p*) = *p*+*1* **by** *simp*
qed

lemma *sigma-third-divisor*:
assumes *1* < *a a* < *n a* : *divisors n*
shows *1*+*a*+*n* <= *sigma*(*n*)
proof −
from *assms* **have** *finite* {*1*,*a*,*n*} & *finite* (*divisors n*) & {*1*,*a*,*n*} <= *divisors n*
by *auto*
hence ∑ {*1*,*a*,*n*} <= ∑ (*divisors n*) **by** (*simp only*: *sum-mono2*)
hence ∑ {*1*,*a*,*n*} <= *sigma n* **by** (*simp add*: *sigma-divisors*)
with *assms* **show** *?thesis* **by** *auto*
qed

lemma *sigma-imp-divisors*: *sigma*(*n*)=*n*+*1* ==> *n*>*1* & *divisors n* = {*n*,*1*}
proof
assume *ass*:*sigma*(*n*)=*n*+*1*
hence *n*≠*0* & *n*≠*1*
by (*metis Suc-eq-plus1 n-not-Suc-n sigma0 sigma1*)
thus *conc1*: *n*>*1* **by** *simp*

show *divisors n* = {*n*,*1*}
proof (*rule ccontr*)
assume *divisors n* ≠ {*n*,*1*}
with *conc1* **have** *divisors n* ≠ {*n*,*1*} & *1*<*n* **by** *auto*
moreover
from *ass conc1* **have** *1* : *divisors*(*n*) & *n* : *divisors n* & ~*0* : *divisors n*
by (*simp add*: *dvd-def divisors-def*)
ultimately
have (∃ *a*. *a*≠*n* & *a*≠*1* & *1*<*n* & *a* : *divisors n*) & *0* ~: *divisors n* **by** *auto*
hence (∃ *a*. *a*≠*n* & *a*≠*1* & *1*<*n* & *a*≠*0* & *a* : *divisors n*) **by** *metis*
hence ∃ *a* . *a*≠*n* & *a*≠*1* & *1*≠*n* & *a*≠*0* & *finite* {*1*,*a*,*n*} & *finite* (*divisors n*)
& {*1*,*a*,*n*} <= *divisors n* **by** *auto*
hence ∃ *a*. *a*≠*n* & *a*≠*1* & *1*≠*n* & *a*≠*0* & ∑ {*1*,*a*,*n*} <= *sigma n*
by (*metis sum-mono2-nat sigma-divisors*)
hence ∃ *a*. *a*≠*0* & (*1*+*a*+*n*) <= *sigma n* **by** *auto*
hence *1*+*n*<*sigma n* **by** *auto*
with *ass* **show** *False* **by** *auto*
qed
qed

lemma *sigma-imp-prime*: *sigma(n)=n+1 ==> prime n*
proof −
assume *ass*: *sigma(n)=n+1*
hence *n>1 & divisors(n)={1,n}* **by** (*metis insert-commute sigma-imp-divisors*)
thus *prime n* **by** (*simp add*: *prime-divisors*)
qed

lemma *pr-pow-div-eq-sm-pr-pow*:
fixes *p*::*nat*
assumes *prime*: *prime p*
shows {*d . d dvd p^n*} = {*p^f| f . f<=n*}
proof
show {*p^f | f . f<=n*} <= { *d . d dvd p^n*}
proof
fix *x*
assume *x*: {*p ^ f | f . f <= n*}
hence ∃ *i . x = p^i & i<= n* **by** *auto*
with *prime* **have** *x dvd p^n*
by (*metis le-imp-power-dvd*)
thus *x* : { *d . d dvd p^n*} **by** *auto*
qed
next
show {*d. d dvd p ^ n*} <= {*p ^ f | f . f <= n*}
proof
fix *x*
assume *x* : {*d . d dvd p^n*}
hence *x dvd p^n* **by** *auto*
with *prime* **obtain** *i* **where** *i <= n & x = p^i* **using** *prime-dvd-power-nat-iff prime-dvd-power-nat*
by (*auto simp only*: *divides-primepow-nat*)
hence *x = p^i & i <=n* **by** *auto*
thus *x* : { *p^f | f . f<=n* } **by** *auto*
qed
qed

lemma *rewrite-sum-of-powers*:
assumes *p*: (*p*::*nat*)*>1*
shows (∑ {*p^m | m . m<=*(*n*::*nat*)}) = (∑ *i = 0 .. n . p^i*) (**is** *?l = ?r*)
proof −
have *?l = sum* (%*x. x*) {((^) *p*) *m* |*m . m<= n*} **by** *auto*
also have ... = *sum* (%*x. x*) (((^) *p*)‘{*m . m<= n*})
by (*simp add*: *setcompr-eq-image*)
moreover with *p* **have** *inj-on* ((^) *p*) {*m . m<=n*}
by (*simp add*: *inj-on-def*)
ultimately have *?l = sum* ((^) *p*) {*m . m<=n*}
by (*simp add*: *sum.reindex*)
moreover have {*m*::*nat . m<=n*} = {*0..n*} **by** *auto*
ultimately show *?l* = (∑ *i = 0 .. n . p^i*) **by** *auto*
qed

theorem *sigma-primepower*:
prime p ==> (p − 1)∗sigma(p^(e::nat)) = (p^(e+1) − 1)
proof −
assume *prime p*
hence *sigma(p^(e::nat)) = (∑ i=0 .. e . p^i)*
by (*simp add*: *pr-pow-div-eq-sm-pr-pow sigma-def rewrite-sum-of-powers prime-nat-iff*)
thus *(p − 1)∗sigma(p^e)=p^(e+1) − 1* **by** (*simp only*: *simplify-sum-of-powers*)
qed

lemma *sigma-prime-power-two*: *sigma(2^(n::nat)) = 2^(n+1) − 1*
proof −
have *(2 − 1)∗sigma(2^(n::nat))=2^(n+1) − 1*
by (*auto simp only*: *sigma-primepower two-is-prime-nat*)
thus *?thesis* **by** *simp*
qed

lemma *prodsums-eq-sumprods*:
fixes *p* :: *nat* **and** *m* :: *nat*
assumes *coprime p m*
shows $\sum\{p\ \hat{}\ f\ |f.\ f \leq n\} * \sum\{b.\ b\ dvd\ m\} = \sum\{p\ \hat{}\ f * b\ |f\ b.\ f \leq n \wedge b\ dvd\ m\}$
proof −
have *coprime p x* **if** *x dvd m* **for** *x*
using *assms* **by** (*rule coprime-imp-coprime*) (*auto intro*: *dvd-trans that*)
then have *coprime (p ^ f) x* **if** *x dvd m* **for** *x f*
using *that* **by** *simp*
then show *?thesis*
by (*auto simp*: *imp-ex sum-mult-sum-if-inj* [*OF mult-inj-if-coprime-nat*]
intro!: *arg-cong* [**where** *f* = *sum* (λ*x*. *x*)])
qed

declare [[*simproc add*: *finite-Collect*]]

lemma *rewrite-for-sigma-semimultiplicative*:
fixes *p*::*nat*
assumes *prime p*
shows *{p^f∗b |f b. f<=n & b dvd m} = {a∗b |a b. a dvd (p^n) & b dvd m}*
proof
show *{p^f ∗ b |f b. f <= n & b dvd m} <= {a∗b |a b. a dvd p ^ n & b dvd m}*
proof
fix *x*
assume *x : {p ^ f ∗ b | f b. f <= n & b dvd m}*
then obtain *b f* **where** *x = p^f∗b & f <= n & b dvd m* **by** *auto*
with ‹*prime p*› **show** *x : {a ∗ b |a b. a dvd p ^ n & b dvd m}*
by (*auto simp add*: *divides-primepow-nat*)
qed
next
show *{a∗b |a b. a dvd p ^ n & b dvd m} <= {p^f ∗ b |f b. f <= n & b dvd m}*

using ‹*prime p*› **by** *auto* (*metis assms divides-primepow-nat*)
qed

lemma *div-decomp-comp*:
fixes *a*::*nat*
shows *coprime m n* ⟹ *a dvd m*∗*n* ⟷ (∃ *b c*. *a* = *b* ∗ *c* & *b dvd m* & *c dvd n*)
by (*auto simp only*: *division-decomp mult-dvd-mono*)

theorem *sigma-semimultiplicative*:
assumes *p*: *prime p* **and** *cop*: *coprime p m*
shows *sigma* (*p* ^ *n*) ∗ *sigma m* = *sigma* (*p* ^ *n* ∗ *m*) (**is** *?l* = *?r*)
proof −
from *cop* **have** *cop2*: *coprime* (*p* ^ *n*) *m*
by *simp*
have *?l* = (∑ {*a* . *a dvd p* ^ *n*})∗(∑ {*b* . *b dvd m*}) **by** (*simp add*: *sigma-def*)
also from *p* **have** ... = (∑ {*p* ^ *f*| *f* . *f*<=*n*})∗(∑ {*b* . *b dvd m*})
by (*simp add*: *pr-pow-div-eq-sm-pr-pow*)
also from *cop* **have** ... = (∑ {*p* ^ *f*∗*b*| *f b* . *f*<=*n* & *b dvd m*})
by (*auto simp add*: *prodsums-eq-sumprods prime-nat-iff*)
also have ... = (∑ {*a*∗*b*| *a b* . *a dvd* (*p* ^ *n*) & *b dvd m*})
by (*simp add*: *p rewrite-for-sigma-semimultiplicative*)
finally have *?l* = ∑{*c*. *c dvd* (*p* ^ *n*∗*m*)} **by** (*subst div-decomp-comp*[*OF cop2*])
thus *?l* = *sigma* (*p* ^ *n*∗*m*) **by** (*auto simp add*: *sigma-def*)
qed

end

3 Perfect Number Theorem

theory *Perfect*
imports *Sigma*
begin

definition *perfect* :: *nat* => *bool* **where**
perfect m == *m*>*0* & *2*∗*m* = *sigma m*

theorem *perfect-number-theorem*:
assumes *even*: *even m* **and** *perfect*: *perfect m*
shows ∃ *n* . *m* = *2* ^ *n*∗(*2* ^ (*n*+*1*) − *1*) ∧ *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-geI*)

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 & 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) >= 2^2* **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** (*rule mult-divisors2*)
ultimately have *1+?B+?A <= sigma ?A* **by** (*rule sigma-third-divisor*)
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 only*: *add-mult-distrib-three mult.commute*)
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*: *sigma-imp-prime*)
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*))
proof (*unfold perfect-def*, *auto*)
from *assms* **show** (*2*::*nat*)∗*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/.