Amicable Numbers

Angeliki Koutsoukou-Argyraki

May 26, 2024

Abstract

This is a formalisation of Amicable Numbers, involving some relevant material including Euler's sigma function, some relevant definitions, results and examples as well as rules such as Thābit ibn Qurra's Rule, Euler's Rule, te Riele's Rule and Borho's Rule with breeders.

The main sources are [2] [3]. Some auxiliary material can be found in [1] [4]. If not otherwise stated, the source of definitions is [2]. In a few definitions where we refer to Wikipedia articles [5] [6] [7] this is explicitly mentioned.

Contents

theory *Amicable-Numbers*
imports *HOL−Number-Theory.Number-Theory*
HOL−Computational-Algebra.Computational-Algebra
Pratt-Certificate.Pratt-Certificate-Code
Polynomial-Factorization.Prime-Factorization

begin

1 Miscellaneous

lemma *mult-minus-eq-nat*:
fixes $x::nat$ **and** $y::nat$ **and** $z::nat$
assumes $x+y = z$
shows $-x-y = -z$
using *assms* **by** *linarith*

lemma *minus-eq-nat-subst*: **fixes** $A::nat$ **and** $B::nat$ **and** $C::nat$ **and** $D::nat$ **and** $E::nat$
assumes $A = B-C-D$ **and** $-E = -C-D$
shows $A = B-E$
using *assms* **by** *linarith*

lemma *minus-eq-nat-subst-order*: **fixes** $A::nat$ **and** $B::nat$ **and** $C::nat$ **and** $D::nat$ **and** $E::nat$
assumes $B-C-D > 0$ **and** $A = B-C-D+B$ **shows** $A = 2*B-C-D$
using *assms* **by** *auto*

lemma *auxiliary-ineq*: **fixes** $x::nat$ **assumes** $x \geq (2::nat)$
shows $x+1 < (2::nat)*x$
using *assms* **by** *linarith*

lemma *sum-strict-mono*:
fixes $A :: nat\ set$
assumes *finite* B $A \subset B$ $0 \notin B$
shows $\sum A < \sum B$
proof −
have $B - A \neq \{\}$
using *assms*(2) **by** *blast*
with *assms DiffE* **have** $\sum (B-A) > 0$
by *fastforce*
moreover have $\sum B = \sum A + \sum (B-A)$
by (*metis add.commute assms*(1) *assms*(2) *psubsetE sum.subset-diff*)
ultimately show *?thesis*
by *linarith*
qed

lemma *coprime-dvd-aux*:

assumes *gcd m n* = *Suc 0 na dvd n ma dvd m mb dvd m nb dvd n* **and** *eq*: *ma* ∗ *na* = *mb* ∗ *nb*
shows *ma* = *mb*
proof −
have *gcd na mb* = *1*
using *assms* **by** (*metis One-nat-def gcd.commute gcd-nat.mono is-unit-gcd-iff*)
moreover have *gcd nb ma* = *1*
using *assms* **by** (*metis One-nat-def gcd.commute gcd-nat.mono is-unit-gcd-iff*)
ultimately show *ma* = *mb*
by (*metis eq gcd-mult-distrib-nat mult.commute nat-mult-1-right*)
qed

2 Amicable Numbers

2.1 Preliminaries

definition *divisor* :: *nat* ⇒*nat* ⇒ *bool* (**infixr** *divisor 80*)
where *n divisor m* ≡(*n* ≥ *1* ∧ *n* ≤ *m* ∧ *n dvd m*)

definition *divisor-set*: *divisor-set m* = {*n*. *n divisor m*}

lemma *def-equiv-divisor-set*: *divisor-set* (*n*::*nat*) = *set*(*divisors-nat n*)
using *divisors-nat-def divisors-nat divisor-set divisor-def* **by** *auto*

definition *proper-divisor* :: *nat* ⇒*nat* ⇒ *bool* (**infixr** *properdiv 80*)
where *n properdiv m* ≡(*n* ≥ *1* ∧ *n* < *m* ∧ *n dvd m*)

definition *properdiv-set*: *properdiv-set m* = {*n*. *n properdiv m*}

lemma *example1-divisor*: **shows** (*2*::*nat*) ∈ *divisor-set* (*4*::*nat*)
using *divisor-set divisor-def* **by** *force*

lemma *example2-properdiv-set*: *properdiv-set* (*Suc* (*Suc* (*Suc 0*))) = {(*1*::*nat*)}
by (*auto simp*: *properdiv-set proper-divisor-def less-Suc-eq dvd-def*; *presburger*)

lemma *divisor-set-not-empty*: **fixes** *m*::*nat* **assumes** *m* ≥*1*
shows *m* ∈ *divisor-set m*
using *assms divisor-set divisor-def* **by** *force*

lemma *finite-divisor-set* [*simp*]: *finite*(*divisor-set n*)
using *divisor-def divisor-set* **by** *simp*

lemma *finite-properdiv-set*[*simp*]: **shows** *finite*(*properdiv-set m*)
using *properdiv-set proper-divisor-def* **by** *simp*

lemma *divisor-set-mult*:
divisor-set (*m*∗*n*) = {*i*∗*j*| *i j*. (*i* ∈ *divisor-set m*)∧(*j* ∈ *divisor-set n*)}
using *divisor-set divisor-def*
by (*fastforce simp add*: *divisor-set divisor-def dest*: *division-decomp*)

lemma *divisor-set-1* [*simp*]: *divisor-set* (*Suc 0*) = {*Suc 0*}
by (*simp add*: *divisor-set divisor-def cong*: *conj-cong*)

lemma *divisor-set-one*: **shows** *divisor-set 1* ={*1*}
using *divisor-set divisor-def* **by** *auto*

lemma *union-properdiv-set*: **assumes** $n \geq 1$ **shows** *divisor-set n* =(*properdiv-set n*)∪{*n*}
using *divisor-set properdiv-set proper-divisor-def assms divisor-def* **by** *auto*

lemma *prime-div-set*: **assumes** *prime n* **shows** *divisor-set n* = {*n*, *1*}
using *divisor-def assms divisor-set prime-nat-iff* **by** *auto*

lemma *div-set-prime*:
assumes *prime n*
shows *properdiv-set n* = {*1*}
using *assms properdiv-set prime-nat-iff proper-divisor-def*
by (*metis* (*no-types*, *lifting*) *Collect-cong One-nat-def divisor-def divisor-set divisor-set-one*
dvd-1-left empty-iff insert-iff mem-Collect-eq order-less-irrefl)

lemma *prime-gcd*: **fixes** *m*::*nat* **and** *n*::*nat* **assumes** *prime m* **and** *prime n* **and** $m \neq n$ **shows** *gcd m n* =*1* **using** *prime-def*
by (*simp add*: *assms primes-coprime*)

We refer to definitions from [5]:

definition *aliquot-sum* :: *nat* ⇒ *nat*
where *aliquot-sum n* ≡ $\sum$ (*properdiv-set n*)

definition *deficient-number* :: *nat* ⇒ *bool*
where *deficient-number n* ≡ (*n* > *aliquot-sum n*)

definition *abundant-number* :: *nat* ⇒ *bool*
where *abundant-number n* ≡ (*n* < *aliquot-sum n*)

definition *perfect-number* :: *nat* ⇒ *bool*
where *perfect-number n* ≡ (*n* = *aliquot-sum n*)

lemma *example-perfect-6*: **shows** *perfect-number 6*

proof−
have *a*: *set*(*divisors-nat 6*) = {*1*, *2*, *3*, *6*} **by** *eval*
have *b*: *divisor-set* (*6*) = {*1*, *2*, *3*, *6*}
using *a def-equiv-divisor-set* **by** *simp*
have *c*: *properdiv-set* (*6*) = {*1*, *2*, *3*}
using *b union-properdiv-set properdiv-set proper-divisor-def* **by** *auto*
show *?thesis* **using** *aliquot-sum-def c*
by (*simp add*: *numeral-3-eq-3 perfect-number-def*)

qed

2.2 Euler's sigma function and properties

The sources of the following useful material on Euler's sigma function are [2], [3], [4] and [1].

definition *Esigma* :: *nat* ⇒ *nat*
  where *Esigma n* ≡ $\sum$ (*divisor-set n*)

lemma *Esigma-properdiv-set*:
  assumes *m* ≥ *1*
  shows *Esigma m* = (*aliquot-sum m*) + *m*
  using *assms divisor-set properdiv-set proper-divisor-def union-properdiv-set Esigma-def*
    aliquot-sum-def **by** *fastforce*

lemma *Esigmanotzero*:
  assumes *n* ≥ *1*
  shows *Esigma n* ≥ *1*
  using *Esigma-def assms Esigma-properdiv-set* **by** *auto*

lemma *prime-sum-div*:
  assumes *prime n*
  shows *Esigma n* = *n* +(*1*::*nat*)
proof −
  have *1* ≤ *n*
    using *assms prime-ge-1-nat* **by** *blast*
  then show *?thesis* **using** *Esigma-properdiv-set assms div-set-prime*
    by (*simp add*: *Esigma-properdiv-set aliquot-sum-def assms div-set-prime*)
qed

lemma *sum-div-is-prime*:
  assumes *Esigma n* = *n* +(*1*::*nat*) **and** *n* ≥*1*
  shows *prime n*

proof (*rule ccontr*)
  assume *F*: ¬ (*prime n*)
  have *n divisor n* **using** *assms divisor-def* **by** *simp*
  have (*1*::*nat*) *divisor n***using** *assms divisor-def* **by** *simp*

  have *n* ≠ *Suc 0*
    using *Esigma-def assms*(*1*) **by** *auto*
  then have *r*: ∃(*m*::*nat*). *m* ∈ *divisor-set n* ∧ *m*≠ (*1*::*nat*) ∧ *m* ≠ *n*
    using *assms F*
    apply (*clarsimp simp add*: *Esigma-def divisor-set divisor-def prime-nat-iff*)
    by (*meson Suc-le-eq dvd-imp-le dvd-pos-nat*)

  have *Suc n* = $\sum$ {*n*,*1*}
    by (*simp add*: ‹*n* ≠ *Suc 0*›)

```
  moreover
  have divisor-set n ⊃ {n,1}
    using assms divisor-set r ‹1 divisor n› divisor-set-not-empty by auto
  then have ∑(divisor-set n) > ∑{n,1}
     apply (rule sum-strict-mono [OF finite-divisor-set])
    by (simp add: divisor-def divisor-set)
  ultimately
  show False
    using Esigma-def assms(1) by presburger
qed

lemma Esigma-prime-sum:
  fixes k:: nat assumes prime m k ≥1
  shows Esigma (m^k) =( m^(k+(1::nat)) −(1::nat)) /(m−1)

proof−
  have m > 1
    using ‹prime m› prime-gt-1-nat by blast

  have A:  Esigma (m^k) =( ∑ j= 0..k.( m^j))
  proof−
    have AA: divisor-set (m^k) = (λj. m ^ j) ‘ {0..k}
      using assms prime-ge-1-nat
     by (auto simp add: power-increasing prime-ge-Suc-0-nat divisor-set divisor-def
image-iff
 divides-primepow-nat)

    have §: ∑ ((λj. m ^ j) ‘ {..k}) = sum (λj. m ^ j) {0..k}  for k
    proof (induction k)
      case (Suc k)
      have [simp]: m ∗ m ^ k ∉ (^) m ‘ {..k}
     by (metis Suc-n-not-le-n ‹1 < m› atMost-iff image-iff power-Suc power-inject-exp)
      then show ?case
        by (clarsimp simp: atMost-Suc Suc)
    qed auto
    show ?thesis
      by (metis § AA Esigma-def atMost-atLeast0)
  qed
  have B: (∑ i≤k.( m^i)) = ( m^Suc k −(1::nat)) /(m−(1::nat))
    using assms ‹m > 1› Set-Interval.geometric-sum  [of m Suc k]
    apply simp
   by (metis One-nat-def lessThan-Suc-atMost nat-one-le-power of-nat-1 of-nat-diff
of-nat-mult
of-nat-power one-le-mult-iff prime-ge-Suc-0-nat sum.lessThan-Suc)
  show ?thesis using A B assms
    by (metis Suc-eq-plus1 atMost-atLeast0 of-nat-1 of-nat-diff prime-ge-1-nat)
qed

lemma prime-Esigma-mult: assumes prime m and prime n and m ≠ n
```

shows *Esigma* $(m*n) = (Esigma\ n)*(Esigma\ m)$

proof−
have *m divisor* $(m*n)$ **using** *divisor-def assms*
by (*simp add*: *dvd-imp-le prime-gt-0-nat*)
moreover have $\neg(\exists\ k::nat.\ k\ divisor\ (m*n) \wedge k\neq(1::nat)\wedge k \neq m \wedge k \neq n \wedge k\neq m*n)$
using *assms* **unfolding** *divisor-def*
by (*metis One-nat-def division-decomp nat-mult-1 nat-mult-1-right prime-nat-iff*)
ultimately have *c*: *divisor-set* $(m*n) = \{m,\ n,\ m*n,\ 1\}$
using *divisor-set assms divisor-def* **by** *auto*
obtain $m\neq 1$ $n\neq 1$
using *assms not-prime-1* **by** *blast*
then have *dd*: *Esigma* $(m*n) = m + n + m * n + 1$
using *assms* **by** (*simp add*: *Esigma-def c*)
then show *?thesis*
using *prime-sum-div assms* **by** *simp*
qed

lemma *gcd-Esigma-mult*:
assumes *gcd m n = 1*
shows *Esigma* $(m*n) = (Esigma\ m)*(Esigma\ n)$

proof−
have *Esigma* $(m*n) = \sum\ \{i*j|\ i\ j.\ i \in divisor\text{-}set\ m \wedge j \in divisor\text{-}set\ n\}$
by (*simp add*: *divisor-set-mult Esigma-def*)
also have ... $= (\sum i \in divisor\text{-}set\ m.\ \sum j \in divisor\text{-}set\ n.\ i*j)$
proof−
have *inj-on* $(\lambda(i,j).\ i*j)$ $(divisor\text{-}set\ m \times divisor\text{-}set\ n)$
using *assms*
apply (*simp add*: *inj-on-def divisor-set divisor-def*)
by (*metis assms coprime-dvd-aux mult-left-cancel not-one-le-zero*)
moreover have $\{i*j|\ i\ j.\ i \in divisor\text{-}set\ m \wedge j \in divisor\text{-}set\ n\} = (\lambda(i,j).\ i*j)$ ` $(divisor\text{-}set\ m \times divisor\text{-}set\ n)$
by *auto*
ultimately show *?thesis*
by (*simp add*: *sum.cartesian-product sum.image-eq*)
qed
also have ... $= \sum (\ divisor\text{-}set\ m)* \sum (\ divisor\text{-}set\ n)$
by (*simp add*: *sum-product*)
also have ... $= Esigma\ m * Esigma\ n$
by (*simp add*: *Esigma-def*)
finally show *?thesis* **.**
qed

lemma *deficient-Esigma*:
assumes *Esigma* $m < 2*m$ **and** $m \geq 1$
shows *deficient-number m*
using *Esigma-properdiv-set assms deficient-number-def* **by** *auto*

lemma *abundant-Esigma*:
assumes $Esigma\ m > 2*m$ **and** $m \geq 1$
shows *abundant-number m*
using *Esigma-properdiv-set assms abundant-number-def* **by** *auto*

lemma *perfect-Esigma*:
assumes $Esigma\ m = 2*m$ **and** $m \geq 1$
shows *perfect-number m*
using *Esigma-properdiv-set assms perfect-number-def* **by** *auto*

2.3 Amicable Numbers; definitions, some lemmas and examples

definition *Amicable-pair* :: $nat \Rightarrow nat \Rightarrow bool$ (**infixr** *Amic 80*)
where $m\ Amic\ n \equiv ((m = aliquot\text{-}sum\ n) \wedge (n = aliquot\text{-}sum\ m))$

lemma *Amicable-pair-sym*: **fixes** $m::nat$ **and** $n::nat$
assumes *m Amic n* **shows** *n Amic m*
using *Amicable-pair-def assms* **by** *blast*

lemma *Amicable-pair-equiv-def*:
assumes (*m Amic n*) **and** $m \geq 1$ **and** $n \geq 1$
shows $(Esigma\ m = Esigma\ n) \wedge (Esigma\ m = m+n)$
using *assms Amicable-pair-def*
by (*metis Esigma-properdiv-set add.commute*)

lemma *Amicable-pair-equiv-def-conv*:
assumes $m \geq 1$ **and** $n \geq 1$ **and** $(Esigma\ m = Esigma\ n) \wedge (Esigma\ m = m+n)$
shows (*m Amic n*)
using *assms Amicable-pair-def Esigma-properdiv-set*
by (*metis add-right-imp-eq add.commute*)

definition *typeAmic* :: $nat \Rightarrow nat \Rightarrow nat\ list$
where *typeAmic n m* =
$[(card\ \{i.\ \exists\ N.\ n = N*(gcd\ n\ m) \wedge prime\ i \wedge i\ dvd\ N \wedge \neg\ i\ dvd\ (gcd\ n\ m)\}),$
$(card\ \{j.\ \exists\ M.\ m = M*(gcd\ n\ m) \wedge prime\ j \wedge j\ dvd\ M \wedge \neg\ j\ dvd\ (gcd\ n\ m)\})]$

lemma *Amicable-pair-deficient*: **assumes** $m > n$ **and** *m Amic n*
shows *deficient-number m*
using *assms deficient-number-def Amicable-pair-def* **by** *metis*

lemma *Amicable-pair-abundant*: **assumes** $m > n$ **and** *m Amic n*
shows *abundant-number n*
using *assms abundant-number-def Amicable-pair-def* **by** *metis*

lemma *even-even-amicable*: **assumes** *m Amic n* **and** $m \geq 1$ **and** $n \geq 1$ **and** *even m* **and** *even n*
shows $(2*m \neq n)$

proof(*rule ccontr*)
have *a*: *Esigma m* = *Esigma n* **using** ‹*m Amic n*› *Amicable-pair-equiv-def Amicable-pair-def*
assms **by** *blast*

assume ¬ (*2*∗*m* ≠ *n*)
have (*2*∗*m* = *n*) **using** ‹¬ (*2*∗*m* ≠ *n*)› **by** *simp*
have *d*:*Esigma n* = *Esigma* (*2*∗*m*) **using** ‹¬ (*2*∗*m* ≠ *n*)› **by** *simp*

then show *False*
proof−
have *w*: *2*∗*m* ∈ *divisor-set* (*2*∗*m*) **using** *divisor-set assms divisor-set-not-empty*
by *auto*
have *w1*: *2*∗*m* ∉ *divisor-set* (*m*) **using** *divisor-set assms*
by (*simp add*: *divisor-def*)
have *w2*: ∀ *n*::*nat*. *n divisor m* ⟶ *n divisor* (*2*∗*m*)
using *assms divisor-def* **by** *auto*
have *w3*: *divisor-set* (*2*∗*m*) ⊃ *divisor-set m* **using** *divisor-set divisor-def assms w w1 w2*
by *blast*
have *v*: (∑ *i* ∈ (*divisor-set* (*2*∗*m*)).*i*)> (∑ *i* ∈ (*divisor-set m*).*i*)
using *w3 sum-strict-mono* **by** (*simp add*: *divisor-def divisor-set*)
show *?thesis* **using** *v d Esigma-def a* **by** *auto*
qed
qed

2.3.1 Regular Amicable Pairs

definition *regularAmicPair* :: *nat* ⇒ *nat* ⇒ *bool* **where**
regularAmicPair n m ⟷ (*n Amic m* ∧
(∃ *M N g*. *g* = *gcd m n* ∧ *m* = *M*∗*g* ∧ *n* = *N*∗*g* ∧ *squarefree M* ∧
squarefree N ∧ *gcd g M* = *1* ∧ *gcd g N* = *1*))

lemma *regularAmicPair-sym*:
assumes *regularAmicPair n m* **shows** *regularAmicPair m n*

proof−
have *gcd m n* = *gcd n m*
by (*metis* (*no-types*) *gcd.commute*)
then show *?thesis*
using *Amicable-pair-sym assms regularAmicPair-def* **by** *auto*
qed

definition *irregularAmicPair* :: *nat* ⇒ *nat* ⇒ *bool* **where**
irregularAmicPair n m ⟷ ((*n Amic m*) ∧ ¬ *regularAmicPair n m*)

lemma *irregularAmicPair-sym*:
assumes *irregularAmicPair n m*

shows *irregularAmicPair m n*
using *irregularAmicPair-def regularAmicPair-sym Amicable-pair-sym assms* **by** *blast*

2.3.2 Twin Amicable Pairs

We refer to the definition in [6]:

definition *twinAmicPair* :: *nat* ⇒ *nat* ⇒ *bool* **where**
twinAmicPair n m ⟷
$(n\ Amic\ m) \wedge (\neg(\exists k\ l.\ k > Min\ \{n, m\} \wedge k < Max\ \{n, m\} \wedge k\ Amic\ l))$

lemma *twinAmicPair-sym*:
assumes *twinAmicPair n m*
shows *twinAmicPair m n*
using *assms twinAmicPair-def Amicable-pair-sym assms* **by** *auto*

2.3.3 Isotopic Amicable Pairs

A way of generating an amicable pair from a given amicable pair under certain conditions is given below. Such amicable pairs are called Isotopic [2].

lemma *isotopic-amicable-pair*:
fixes *m n g h M N* :: *nat*
assumes *m Amic n* **and** $m \geq 1$ **and** $n \geq 1$ **and** $m = g*M$ **and** $n = g*N$
and $Esigma\ h = (h/g) * Esigma\ g$ **and** $h \neq g$ **and** $h > 1$ **and** $g > 1$
and $gcd\ g\ M = 1$ **and** $gcd\ g\ N = 1$ **and** $gcd\ h\ M = 1$ **and** $gcd\ h\ N = 1$
shows $(h*M)$ *Amic* $(h*N)$

proof−
have *a*: $Esigma\ m = Esigma\ n$ **using** ‹ *m Amic n*› *Amicable-pair-equiv-def assms*
by *blast*
have *b*: $Esigma\ m = m + n$ **using** ‹ *m Amic n*› *Amicable-pair-equiv-def assms*
by *blast*
have *c*: $Esigma\ (h*M) = (Esigma\ h)*(Esigma\ M)$

proof−
have $h \neq M$
using *assms Esigmanotzero gcd-Esigma-mult gcd-nat.idem b mult-eq-self-implies-10*
by (*metis less-irrefl*)

show *?thesis* **using** ‹$h \neq M$› *gcd-Esigma-mult assms*
by *auto*
qed

have *d*: $Esigma\ (g*M) = (Esigma\ g)*(Esigma\ M)$

proof−

```
      have  g≠M  using assms gcd-nat.idem by (metis less-irrefl)
     show ?thesis using ‹g≠M› gcd-Esigma-mult assms by auto
  qed

  have e: Esigma (g*N) = (Esigma g)*(Esigma N)

    proof−
    have  g≠N  using assms by auto
    show ?thesis using ‹g≠N› gcd-Esigma-mult assms by auto
    qed

  have p1: Esigma m = (Esigma g)*(Esigma M) using assms d by simp
  have p2: Esigma n = (Esigma g)*(Esigma N) using assms e by simp
  have p3: Esigma (h*N) = (Esigma h)*(Esigma N)

  proof−
    have  h≠N  using assms ‹ gcd h N =1› a b p2 by fastforce
    show ?thesis using ‹h ≠ N› gcd-Esigma-mult  assms by auto
  qed

  have A: Esigma (h*M) = Esigma (h*N)
    using c p3 d e p1 p2 a assms Esigmanotzero by fastforce

  have B:  Esigma (h*M)=(h*M)+(h*N)
  proof−
     have s: Esigma (h*M) = (h/g)*(m+n) using b c p1 Esigmanotzero assms by
simp
      have s1: Esigma (h*M) = h*(m/g+n/g) using  s assms
        by (metis add-divide-distrib b of-nat-add semiring-normalization-rules(7)
 times-divide-eq-left times-divide-eq-right)
      have s2:  Esigma (h*M) = h*(M+N)
      proof−
        have v: m/g = M using assms by simp
        have v1:n/g = N using assms by simp
        show ?thesis using s1 v v1 assms
          using of-nat-eq-iff by fastforce
      qed
      show ?thesis using s2 assms
        by (simp add: add-mult-distrib2)
   qed
 show ?thesis using Amicable-pair-equiv-def-conv A B assms one-le-mult-iff  One-nat-def
Suc-leI
    by (metis (no-types, opaque-lifting)  nat-less-le)
qed


lemma isotopic-pair-example1:
  assumes (3^3*5*11*17*227) Amic (3^3*5*23*37*53)
  shows  (3^2*7*13*11*17*227) Amic (3^2*7*13*23*37*53)
```

```
proof−
  obtain m where o1: m = (3::nat)^3*5*11*17*227 by simp
  obtain n where o2: n = (3::nat)^3*5*23*37*53 by simp
  obtain g where o3: g = (3::nat)^3*5 by simp
  obtain h where o4: h = (3::nat)^2*7*13 by simp
  obtain M where o5: M = (11::nat)*17*227 by simp
  obtain N where o6: N = (23::nat)*37*53 by simp
  have prime(3::nat) by simp
  have prime(5::nat) by simp
  have prime(7::nat) by simp
  have prime(13::nat) by simp

  have v: m Amic n using o1 o2 assms by simp
  have v1: m = g*M using o1 o3 o5  by simp
  have v2: n = g*N  using o2 o3 o6 by simp
  have v3: h >0 using o4 by simp
  have w: g >0 using o3 by simp
  have w1: h ≠ g using o4 o3 by simp
  have h = 819 using o4 by simp
  have g = 135 using o3 by simp

    have w2: Esigma h = (h/g)*Esigma g

  proof−
  have B: Esigma h = 1456
    proof−
      have R: set(divisors-nat 819) ={1, 3, 7, 9, 13, 21, 39, 63, 91, 117, 273,
819}
        by eval
      have RR: set( divisors-nat(819)) = divisor-set (819)
        using def-equiv-divisor-set by simp

        show?thesis using Esigma-def RR R ‹ h = 819› divisor-def divisors-nat
divisors-nat-def by auto
     qed

    have C: Esigma g = 240
    proof−
      have G: set(divisors-nat 135) = {1, 3, 5, 9, 15, 27, 45, 135}
        by eval
      have GG:  set(divisors-nat 135) = divisor-set 135
        using  def-equiv-divisor-set by simp

   show ?thesis using G GG Esigma-def  ‹ g = 135›
      properdiv-set proper-divisor-def
     by simp
 qed
  have D: (Esigma h) * g = (Esigma g) * h
```

```
    proof−
    have A: (Esigma h) ∗ g = 196560
        using B o3 by simp
      have AA: (Esigma g) ∗ h = 196560 using C o4  by simp
      show ?thesis using A AA by simp
    qed
    show ?thesis using D
      by (metis mult.commute nat-neq-iff nonzero-mult-div-cancel-right
of-nat-eq-0-iff of-nat-mult times-divide-eq-left w)

 qed

  have w4: gcd g M =1

  proof−
    have coprime g M

    proof−
      have ¬ g dvd M  using o3 o5 by auto
      moreover have  ¬ 3 dvd M using o5 by auto
      moreover have  ¬ 5 dvd M  using o5 by auto
      ultimately show ?thesis using o5 o3
     gcd-nat.absorb-iff2 prime-nat-iff ‹ prime(3::nat)› ‹ prime(5::nat)›
    by (metis coprime-commute
coprime-mult-left-iff prime-imp-coprime-nat prime-imp-power-coprime-nat)
qed
 show ?thesis using ‹coprime g M› by simp
  qed

  have s:  gcd g N =1

  proof−
    have coprime g N

 proof−
      have ¬ g dvd N
        using o3 o6  by auto
      moreover have ¬ 3 dvd N using o6 by auto
      moreover have ¬ 5 dvd N using o6 by auto
    ultimately show ?thesis using o3 gcd-nat.absorb-iff2 prime-nat-iff ‹ prime(3::nat)›
 ‹ prime(5::nat)›
    by (metis coprime-commute
coprime-mult-left-iff prime-imp-coprime-nat prime-imp-power-coprime-nat)
qed
 show ?thesis using ‹coprime g N› by simp
  qed

  have s1: gcd h M =1
```

```
 proof−
   have coprime h M

 proof−
      have ¬ h dvd M  using o4 o5  by auto
      moreover have ¬ 3 dvd M using o5  by auto
      moreover have ¬ 7 dvd M using o5  by auto
      moreover have ¬ 13 dvd M using o5 by auto
        ultimately  show ?thesis using o4  gcd-nat.absorb-iff2 prime-nat-iff ‹
prime(3::nat)›
‹ prime(13::nat)› ‹ prime(7::nat)›

    by (metis coprime-commute
coprime-mult-left-iff prime-imp-coprime-nat prime-imp-power-coprime-nat)
qed

  show ?thesis using ‹coprime h M›  by simp
  qed

  have s2: gcd h N =1

  proof−
    have coprime h N

    proof−
  have ¬ h dvd N using o4 o6  by auto
  moreover have ¬ 3 dvd N using o6  by auto
  moreover have ¬ 7 dvd N using o6  by auto
  moreover have ¬ 13 dvd N using o6 by auto
  ultimately show ?thesis using o4
    gcd-nat.absorb-iff2 prime-nat-iff ‹ prime(3::nat)›‹ prime(13::nat)› ‹ prime(7::nat)›

    by (metis coprime-commute
coprime-mult-left-iff prime-imp-coprime-nat prime-imp-power-coprime-nat)
qed

 show ?thesis using ‹coprime h N›  by simp
  qed

  have s4: (h∗M) Amic (h∗N) using isotopic-amicable-pair v v1 v2 v3 w4 s s1 s2
w w1 w2
   by (metis One-nat-def Suc-leI le-eq-less-or-eq nat-1-eq-mult-iff
num.distinct(3) numeral-eq-one-iff one-le-mult-iff one-le-numeral o3 o4 o5 o6)

  show ?thesis using s4 o4 o5 o6 by simp
qed
```

2.3.4 Betrothed (Quasi-Amicable) Pairs

We refer to the definition in [7]:

definition *QuasiAmicable-pair* :: *nat* ⇒ *nat* ⇒ *bool* (**infixr** *QAmic 80*)
  where *m QAmic n* ⟷ (*m* + *1* = *aliquot-sum n*) ∧ (*n* + *1* = *aliquot-sum m*)

lemma *QuasiAmicable-pair-sym* :
  assumes *m QAmic n* **shows** *n QAmic m*
  using *QuasiAmicable-pair-def assms* **by** *blast*

lemma *QuasiAmicable-example*:
  shows *48 QAmic 75*

proof−
  have *a*: *set*(*divisors-nat 48*) = {*1*, *2*, *3*, *4*, *6*, *8*, *12*, *16*, *24*, *48*} **by** *eval*
  have *b*: *divisor-set* (*48*) = {*1*, *2*, *3*, *4*, *6*, *8*, *12*, *16*, *24*, *48*}
    using *a def-equiv-divisor-set* **by** *simp*
  have *c*: *properdiv-set* (*48*) = {*1*, *2*, *3*, *4*, *6*, *8*, *12*, *16*, *24*}
    using *b union-properdiv-set properdiv-set proper-divisor-def* **by** *auto*
  have *e*: *aliquot-sum* (*48*) = *75*+*1* **using** *aliquot-sum-def c*
    by *simp*
  have *i*: *set*(*divisors-nat 75*) = {*1*, *3*, *5*, *15*, *25*, *75*} **by** *eval*
  have *ii*: *divisor-set* (*75*) = {*1*, *3*, *5*, *15*, *25*, *75*}
    using *i def-equiv-divisor-set* **by** *simp*
  have *iii*: *properdiv-set* (*75*) = {*1*, *3*, *5*, *15*, *25*}
    using *ii union-properdiv-set properdiv-set proper-divisor-def* **by** *auto*
  have *iv*: *aliquot-sum* (*75*) = *48*+*1* **using** *aliquot-sum-def iii*
    by *simp*
  show *?thesis* **using** *e iv QuasiAmicable-pair-def* **by** *simp*
qed

2.3.5 Breeders

definition *breeder-pair* :: *nat* ⇒*nat* ⇒ *bool* (**infixr** *breeder 80*)
  where *m breeder n* ≡ (∃ *x*∈ℕ. *x* > *0* ∧ *Esigma m* = *m* + *n*∗*x* ∧ *Esigma m* = (*Esigma n*)∗(*x*+*1*))

lemma *breederAmic*:
  fixes *x* :: *nat*
  assumes *x* > *0* **and** *Esigma n* = *n* + *m*∗*x* **and** *Esigma n* = *Esigma m* ∗ (*x*+*1*)

      and *prime x* **and** ¬(*x dvd m*)
  shows *n Amic* (*m*∗*x*)

proof−
  have *A*: *Esigma n* = *Esigma* (*m*∗*x*)
  proof−
    have *gcd m x* =*1* **using** *assms gcd-nat.absorb-iff2 prime-nat-iff* **by** *blast*

have *A1*: *Esigma* (*m*∗*x*) = (*Esigma m*)∗(*Esigma x*)
using ‹*gcd m x* =*1*› *gcd-Esigma-mult* **by** *simp*
have *A2*: *Esigma* (*m*∗*x*) = (*Esigma m*)∗(*x*+*1*)
using ‹*prime x*› *prime-Esigma-mult A1*
by (*simp add*: *prime-sum-div*)
show *?thesis* **using** *A2 assms* **by** *simp*
qed

have *B*: *Esigma n* = *n*+*m*∗*x* **using** *assms* **by** *simp*
show *?thesis* **using** *A B Amicable-pair-equiv-def assms*
by (*smt* (*verit*, *del-insts*) *Amicable-pair-equiv-def-conv Esigma-properdiv-set Suc-eq-plus1 Suc-le-eq add-0 add-le-same-cancel2*
add-less-cancel-right dvd-triv-right less-nat-zero-code linorder-not-le mult-Suc-right mult-is-0 mult-le-mono not-gr-zero prime-ge-Suc-0-nat)
qed

2.3.6 More examples

The first odd-odd amicable pair was discovered by Euler [2]. In the following proof, amicability is shown using the properties of Euler's sigma function.

lemma *odd-odd-amicable-Euler*: *69615 Amic 87633*
proof−
have *prime*(*5*::*nat*) **by** *simp*
have *prime*(*17*::*nat*) **by** *simp*
have ¬ (*5*∗*17*)*dvd*((*3*::*nat*)^*2*∗*7*∗*13*) **by** *auto*
have ¬ *5 dvd*((*3*::*nat*)^*2*∗*7*∗*13*) **by** *auto*
have ¬ *17 dvd*((*3*::*nat*)^*2*∗*7*∗*13*) **by** *auto*
have *A1*: *Esigma*(*69615*) = *Esigma*(*3*^*2*∗*7*∗*13*∗*5*∗*17*) **by** *simp*
have *A2*: *Esigma*(*3*^*2*∗*7*∗*13*∗*5*∗*17*) = *Esigma*(*3*^*2*∗*7*∗*13*)∗*Esigma*(*5*∗*17*)

proof−
have *A111*: *coprime* ((*3*::*nat*)^*2*∗*7*∗*13*) ((*5*::*nat*)∗*17*)
using ‹¬ *17 dvd*((*3*::*nat*)^*2*∗*7*∗*13*)› ‹¬ *5 dvd*((*3*::*nat*)^*2*∗*7*∗*13*)› ‹*prime* (*17*::*nat*)›
‹*prime* (*5*::*nat*)› *coprime-commute coprime-mult-left-iff prime-imp-coprime-nat* **by** *blast*

have *gcd* (*3*^*2*∗*7*∗*13*)((*5*::*nat*)∗*17*) =*1*
using *A111 coprime-imp-gcd-eq-1* **by** *blast*
then show *?thesis*
by (*metis* (*no-types*, *lifting*) *gcd-Esigma-mult mult.assoc*)
qed
have *prime* (*7*::*nat*) **by** *simp*
have ¬ *7 dvd* ((*3*::*nat*)^*2*) **by** *simp*
have *prime* (*13*::*nat*) **by** *simp*
have ¬ *13 dvd* ((*3*::*nat*)^*2*∗*7*) **by** *simp*
have *gcd* ((*3*::*nat*)^*2*∗*7*) *13* =*1*
using ‹*prime* (*13*::*nat*)› ‹¬ *13 dvd* ((*3*::*nat*)^*2*∗*7*)› *gcd-nat.absorb-iff2 prime-nat-iff*
by *blast*

```
  then have A3:  Esigma(3^2 * 7*13) = Esigma(3^2*7)*Esigma(13)
    using gcd-Esigma-mult by presburger
  have  gcd ((3::nat)^2) 7 = 1
   using ‹prime (7::nat)› ‹ ¬ 7 dvd ((3::nat)^2 )› gcd-nat.absorb-iff2 prime-nat-iff
    by blast
  then have A4:  Esigma(3^2*7) = Esigma(3^2)* Esigma (7)
    using gcd-Esigma-mult by presburger
  have A5: Esigma(3^2) = 13
  proof−
    have  (3::nat)^2 =9 by auto
    have A55:divisor-set 9 = {1, 3, 9}
    proof−
      have A555: set(divisors-nat (9)) = {1, 3, 9} by eval
      show ?thesis using def-equiv-divisor-set A555 by simp
    qed
    show ?thesis using A55 ‹(3::nat)^2 =9› Esigma-def by simp
  qed
  have prime( 13::nat) by simp
  have A6: Esigma (13) = 14
    using prime-sum-div ‹prime( 13::nat)› by auto
  have prime( 7::nat) by simp
  have A7: Esigma (7) = 8
    using prime-sum-div ‹prime( 7::nat)› by auto
  have prime (5::nat) by simp
  have prime (17::nat) by simp
  have A8: Esigma(5*17) = Esigma(5) * Esigma (17)
    using prime-Esigma-mult ‹prime (5::nat)› ‹prime (17::nat)›
     by (metis arith-simps(2) mult.commute num.inject(2) numeral-eq-iff semir-
ing-norm(83))
 have A9: Esigma(69615) = Esigma(3^2)*Esigma (7) *Esigma (13) * Esigma(5)
* Esigma (17)
    using A1 A2 A3 A4 A5 A6 A7 A8 by (metis mult.assoc)
  have A10: Esigma (5)=6
    using prime-sum-div ‹prime(5::nat)› by auto
  have A11: Esigma (17)=18
    using prime-sum-div ‹prime(17::nat)› by auto
  have AA:  Esigma(69615)=13*8*14*6*18 using A1 A2 A3 A4 A5 A6 A7 A8
A9 A10 A11
    by simp
  have AAA: Esigma(69615) =157248 using AA by simp

  have AA1: Esigma(87633) = Esigma (3^2*7*13*107) by simp
  have prime (107::nat) by eval
  have AA2: Esigma (3^2*7*13*107) = Esigma (3^2*7*13)*Esigma(107)

  proof−
    have ¬ (107::nat) dvd  (3^2*7*13) by auto
    then have gcd  ((3::nat)^2*7*13) 107 =1 using ‹prime (107::nat)›
      using gcd-nat.absorb-iff2 prime-nat-iff by blast
```

then show *?thesis*
by (*meson gcd-Esigma-mult*)
qed
have *AA3*: *Esigma* (*107*) =*108*
using *prime-sum-div* ‹*prime*(*107*::*nat*)› **by** *auto*
have *AA4*: *Esigma*(*3^2∗7∗13*) = *13∗8∗14*
using *A3 A4 A5 A6 A7* **by** *auto*
have *AA5* : *Esigma* (*3^2∗7∗13∗107*) = *13∗8∗14∗108*
using *AA2 AA3 AA4* **by** *auto*
have *AA6*: *Esigma* (*3^2∗7∗13∗107*) = *157248* **using** *AA5* **by** *simp*
have *A*:*Esigma*(*69615*) = *Esigma*(*87633*)
using *AAA AA6 AA5 AA1* **by** *linarith*
have *B*: *Esigma*(*87633*) = *69615* + *87633*
using *AAA* ‹*Esigma 69615* = *Esigma 87633*› **by** *linarith*
show *?thesis* **using** *A B Amicable-pair-def Amicable-pair-equiv-def-conv* **by** *auto*
qed

The following is the smallest odd-odd amicable pair [2]. In the following proof, amicability is shown directly by evaluating the sets of divisors.

lemma *Amicable-pair-example-smallest-odd-odd*: *12285 Amic 14595*
proof−
have *A*: *set*(*divisors-nat* (*12285*)) = {*1*, *3*, *5*, *7*, *9*, *13*, *15*, *21*, *27*, *35*, *39*, *45*, *63*, *65*, *91*,
105, *117*, *135*, *189*, *195*, *273*, *315*, *351*, *455*, *585*, *819*, *945*, *1365*, *1755*, *2457*, *4095*, *12285*}
by *eval*
have *A1*: *set*(*divisors-nat* (*12285*)) = *divisor-set 12285*
using *def-equiv-divisor-set* **by** *simp*
have *A2*: ∑{*1*, *3*, *5*, *7*, *9*, *13*, *15*, *21*, *27*, *35*, *39*, *45*, *63*, *65*, *91*, *105*, *117*, *135*, *189*, *195*, *273*,
315, *351*, *455*, *585*, *819*, *945*, *1365*, *1755*, *2457*, *4095*, *12285*} = (*26880*::*nat*)
by *eval*
have *A3*: *Esigma 12285* = *26880* **using** *A A1 A2 Esigma-def* **by** *simp*
have *Q*:*Esigma 12285* = *Esigma 14595*
proof−
have *N*: *set*(*divisors-nat* (*14595*)) =
{ *1*, *3*, *5*, *7*, *15*, *21*, *35*, *105*, *139*, *417*, *695*, *973*, *2085*, *2919*, *4865*, *14595*}
by *eval*
have *N1*: *set*(*divisors-nat* (*14595*)) = *divisor-set 14595*
using *def-equiv-divisor-set* **by** *simp*
have *N2*:
∑{ *1*, *3*, *5*, *7*, *15*, *21*, *35*, *105*, *139*, *417*, *695*, *973*, *2085*, *2919*, *4865*, *14595*} = (*26880*::*nat*)
by *eval*
show *?thesis* **using** *A3 N N1 N2 Esigma-def* **by** *simp*
qed
have *B*:*Esigma* (*12285*) = *12285* + *14595* **using** *A3* **by** *auto*
show *?thesis* **using** *B Q Amicable-pair-def*

```
    using Amicable-pair-equiv-def-conv one-le-numeral by blast
qed
```

3 Euler's Rule

We present Euler's Rule as in [2]. The proof has been reconstructed.

```
theorem Euler-Rule-Amicable:
  fixes k l f p q r m n :: nat
  assumes k > l and l ≥ 1 and f = 2^l+1
      and prime p and prime q and prime r
       and p = 2^(k−l) ∗ f − 1 and q = 2^k ∗ f − 1 and r = 2^(2∗k−l) ∗ f^2
− 1
      and m = 2^k ∗ p ∗ q and n = 2^k ∗ r
  shows m Amic n

proof−
  note [[linarith-split-limit = 50]]
  have A1: (p+1)∗(q+1) = (r+1)
  proof−
    have a: p+1 = (2^(k−l))∗f using assms by simp
    have b: q+1 = (2^(k))∗f using assms by simp
    have c: r+1 = (2^(2∗k−l))∗(f^2) using assms by simp
    have d: (p+1)∗(q+1) = (2^(k−l))∗(2^(k))∗f^2
      using a b by (simp add: power2-eq-square)
    show ?thesis using d c
       by (metis Nat.add-diff-assoc add.commute assms(1) less-imp-le-nat mult-2
power-add)
  qed
  have aa: Esigma p = p+1 using assms ‹prime p› prime-sum-div by simp
  have bb: Esigma q = q+1 using ‹prime q› prime-sum-div assms by simp
  have cc: Esigma r = r+1 using ‹prime r› prime-sum-div assms by simp
  have A2: (Esigma p)∗(Esigma q) = Esigma r
    using aa bb cc A1 by simp
  have A3: Esigma (2^k)∗(Esigma p)∗(Esigma q) = Esigma(2^k)∗(Esigma r)
    using A2 by simp
  have A4: Esigma(( 2^k)∗r) = (Esigma(2^k))∗(Esigma r)
  proof−
    have Z0: gcd ((2::nat)^k)r =1 using assms ‹prime r›  by simp
    have A: (2::nat)^k ≥ 1 using assms by simp
    have Ab: (2::nat)^k ≠ r using assms
      by (metis gcd-nat.idem numeral-le-one-iff prime-ge-2-nat semiring-norm(69)
Z0)
    show ?thesis using Z0 gcd-Esigma-mult assms A Ab by metis
  qed

  have A5: Esigma((2^k)∗p∗q) =(Esigma(2^k))∗(Esigma p)∗(Esigma q)
  proof−
    have (2::nat)^k ≥1 using assms by simp
```

```
    have A:  gcd (2^k) p =1 using assms ‹prime p› by simp
    have B:  gcd (2^k) q =1 using assms ‹prime q› by simp
    have BB: gcd (2^k) (p*q) =1 using assms A B by fastforce
    have C:  p*q ≥1 using assms One-nat-def one-le-mult-iff prime-ge-1-nat by
metis
    have A6:  Esigma((2^k)*(p*q))=( Esigma(2^k))*(Esigma(p*q))
    proof−
      have (( 2::nat)^k) ≠ (p*q) using assms
        by (metis BB Nat.add-0-right gcd-idem-nat less-add-eq-less
         not-add-less1 power-inject-exp prime-gt-1-nat semiring-normalization-rules(32)
            two-is-prime-nat )
      show ?thesis using ‹(( 2::nat)^k) ≠ (p*q)›
         ‹( 2::nat)^k ≥1›  gcd-Esigma-mult assms C BB
        by metis
    qed
    have A7:Esigma(p*q) = (Esigma p)*(Esigma q)
    proof−
      have p ≠ q
           by (metis A2 Suc-eq-plus1 Zero-not-Suc aa add-Suc add-cancel-left-left
add-diff-cancel-left′ assms(4) assms(6)
           cc diff-is-0-eq linorder-not-less mult.commute mult-Suc-right prime-nat-iff
prime-product)
      show ?thesis using ‹p ≠ q›
         ‹prime p›  ‹prime q› C  prime-Esigma-mult assms
        by (metis mult.commute)
    qed

     have A8: Esigma((2^k)*( p*q))=(Esigma(2^k))*(Esigma p)*(Esigma q) by
(simp add: A6 A7)
    show ?thesis using A8 by (simp add: mult.assoc)
  qed

   have Z: Esigma((2^k)*p*q) = Esigma ((2^k)*r) using A1 A2 A3 A4 A5 by
simp

  have Z1: Esigma ((2^k)*p*q) = 2^k *p*q + 2^k*r

  proof−
    have prime (2::nat) by simp
    have s: Esigma (2^k) =((2::nat)^(k+1)−1)/(2−1)
      using ‹prime (2::nat)› assms Esigma-prime-sum by auto
    have ss: Esigma (2^k) =(2^(k+1)−1) using s by simp
    have J: (k+1+k−l+k)= 3*k +1−l using assms by linarith
    have JJ: (2^(k−l))*(2^k) = (2::nat)^(2*k−l)
      apply (simp add: algebra-simps)
     by (metis Nat.add-diff-assoc assms(1) less-imp-le-nat mult-2-right power-add)
     have Esigma((2^k)*p*q)= (Esigma(2^k))*(Esigma p)*(Esigma q) using A5
by simp
    also have . . . = (2^(k+1)−1)*(p+1)*(q+1) using assms ss aa bb by metis
```

```
also have ... = (2^(k+1)−1)∗((2^(k−l))∗f)∗((2^k)∗f) using assms by simp
also have ... = (2^(k+1)−1)∗(2^(k−l))∗(2^k)∗f^2
  by (simp add: power2-eq-square)
 also have ... = (2^(k+1))∗(2^(k−l))∗(2^k)∗f^2−(2^(k−l))∗(2^k)∗f^2
  using diff-mult-distrib nat-mult-1 by presburger
also have ... = (2^(k+1+k−l+k))∗f^2−(2^(k−l))∗(2^k)∗f^2
  by (metis Nat.add-diff-assoc assms(1) less-imp-le-nat power-add)
also have ... = (2^(3∗k+1−l))∗f^2−(2^(k−l))∗(2^k)∗f^2
 using J by auto
also have ... = (2^(3∗k+1−l))∗f^2−(2^(2∗k−l))∗f^2
 using JJ by simp
finally
have YY: Esigma((2^k)∗p∗q)= (2^(3∗k+1−l))∗f^2−(2^(2∗k−l))∗f^2 .

 have auxicalc: (2^(2∗k−l))∗(f^2)=(2^(2∗k−l))∗f +(2^(2∗k))∗f

 proof−
  have i: (2^(2∗k−l))∗f = (2^(2∗k−l))∗(2^l+1)
   using assms ‹f = 2^l+1› by simp
   have ii: ( 2^(2∗k−l))∗f = (2^(2∗k−l))∗( 2^l)+(2^(2∗k−l))
    using i by simp
   have iii: (2^(2∗k−l))∗f = (2^(2∗k−l+l))+(2^(2∗k−l))
    using ii  by (simp add: power-add)
   have iv: ( 2^(2∗k−l))∗f∗f =(((2^(2∗k))+(2^(2∗k−l))))∗f
    using iii assms by simp
   have v:  (2^(2∗k−l))∗f ∗f =((2^(2∗k)))∗f+((2^(2∗k−l)))∗f
  using iv assms comm-monoid-mult-axioms  power2-eq-square semiring-normalization-rules(18)
      semiring-normalization-rules by (simp add: add-mult-distrib assms)
 show ?thesis using v by (simp add: power2-eq-square semiring-normalization-rules(18))
 qed

 have W1: 2^k∗p∗q + 2^k∗r = 2^k ∗(p∗q +r)
  by (simp add: add-mult-distrib2)

have W2: 2^k∗(p∗q +r)= 2^k ∗((2^(k−l)∗f−1)∗((2^k)∗f−1)+(2^(2∗k−l))∗f^2−1)
  using assms by simp

 have W3: 2^k∗((2^(k−l)∗f−1)∗((2^k)∗f−1)+(2^(2∗k−l))∗f^2−1)=
2^k∗((2^(k−l)∗f−1)∗((2^k)∗f)−(2^(k−l)∗f−1)+(2^(2∗k−l))∗f^2−1)
  by (simp add: right-diff-distrib')

have W4: 2^k∗((2^(k−l)∗f−1)∗((2^k)∗f)−(2^(k−l)∗f−1)+(2^(2∗k−l))∗f^2−1)
=
2^k∗((2^(k−l)∗f)∗((2^k)∗f)−((2^k)∗f)−(2^(k−l)∗f−1)+(2^(2∗k−l))∗f^2−1)
  using assms  by (simp add: diff-mult-distrib)

have W5:  2^k∗((2^(k−l)∗f)∗((2^k)∗f)−((2^k)∗f)−(2^(k−l)∗f−1)+(2^(2∗k−l))∗f^2−1)
=
2^k ∗(( 2^(k−l)∗f)∗((2^k)∗f)−((2^k)∗f)−(2^(k−l)∗f)+1 +(2^(2∗k−l))∗f^2−1)
```

using *assms less-imp-le-nat prime-ge-1-nat*
by (*smt* (*verit*) *Nat.add-diff-assoc2 Nat.diff-diff-right One-nat-def W3 W4 add.commute add-diff-cancel-right'*
add-diff-inverse-nat diff-is-0-eq le-add2 nat-mult-eq-cancel-disj one-le-mult-iff plus-1-eq-Suc)

have *W6*: $2\hat{}k*((2\hat{}(k-l)*f)*((2\hat{}k)*f)-((2\hat{}k)*f)-(2\hat{}(k-l)*f)+1+(2\hat{}(2*k-l))*f\hat{}2-1) =$
$2\hat{}k*((2\hat{}(k-l)*f)*((2\hat{}k)*f)-((2\hat{}k\ )*f)-(2\hat{}(k-l)*f)+(2\hat{}(2*k-l))*f\hat{}2)$
by *simp*

have *W7*: $2\hat{}k*((2\hat{}(k-l)*f)*((2\hat{}k)*f)-((2\hat{}k)*f)-(2\hat{}(k-l)*f)+(2\hat{}(2*k-l))*f\hat{}2) =$
$2\hat{}k\ *((2\hat{}(2*k-l+1)*(f\hat{}2))-((2\hat{}k)*f)-(2\hat{}(k-l)*\ f))$

proof−
have *a*: $(2\hat{}(k-l)*f)*(2\hat{}k\ *\ f) = (2\hat{}(k-l)*f*(f*(2\hat{}k)))$
using *assms* **by** *simp*
have *b*: $(2\hat{}(k-l)*f)*(f*(2\hat{}k)) = 2\hat{}(k-l)*(f*f)*(2\hat{}k)$
using *assms* **by** *linarith*
have *c*: $2\hat{}(k-l)*(f*f)*(2\hat{}k) = 2\hat{}(k-l+k)*(f\hat{}2)$
using *Semiring-Normalization.comm-semiring-1-class.semiring-normalization-rules*(*16*)
Semiring-Normalization.comm-semiring-1-class.semiring-normalization-rules(*29*)
by (*simp add*: *power-add*)

have *d*: $2\hat{}(k-l+k)\ *(f\hat{}2) = 2\hat{}(2*k-l)\ *(f\hat{}2)$
by (*simp add*: *JJ power-add*)

have *e*: $(2\hat{}(2*k-l))*f\hat{}2 + (2\hat{}(2*k-l))*f\hat{}2 = 2\hat{}(2*k-l\ +1)*(f\hat{}2)$
by *simp*

have *f1*: $((2\hat{}(k-l)*f)*((2\hat{}k)*f)-((2\hat{}k)*f)-(2\hat{}(k-l)*f)+(2\hat{}(2*k-l))*f\hat{}2) =$
$(2\hat{}(2*k-l)*(f\hat{}2)-((2\hat{}k)*f)-(2\hat{}(k-l)*f)+(2\hat{}(2*k-l))*f\hat{}2)$
using *a b c d e* **by** *simp*

have *f2*: $((2\hat{}(k-l)*f)*((2\hat{}k)*f)-((2\hat{}k)*f)-(2\hat{}(k-l)*f))+(2\hat{}(2*k-l))*f\hat{}2$
$= ((2\hat{}(2*k-l+1)*(f\hat{}2))-((2\hat{}k)*f)-(2\hat{}(k-l)*f))$

proof−
have *aa*: $f > 1$ **using** *assms* **by** *simp*
have *a*: $((2::nat)\hat{}(2*k-l))*f\hat{}2-((2::nat)\hat{}(k-l)*f)>0$
proof−
have *b*: $(2::nat)\hat{}(2*k-l) > 2\hat{}(k-l)$ **using** *assms* **by** *simp*
have *c*: $(2::nat)\hat{}(2*k-l)*f > 2\hat{}(k-l)*f$ **using** *a assms*
by (*metis One-nat-def add-gr-0 b lessI mult-less-mono1*)
show *?thesis*
using *c auxicalc* **by** *linarith*
qed

```
    have aaa: (2^(2*k−l))*f^2 −(2^(k−l)*f)−((2^k)*f) >0

    proof−
      have A: (2^(2*k−l))*f−(2^(k−l))−(( 2^k)) >0

      proof−
          have A-1 : (2^(2*k−l))*f  > (2^(k−l))+((2^k))

        proof−
          have A-2: (2^(2*k−l))*f = 2^(k)*2^(k−l)*f
              by (metis JJ semiring-normalization-rules(7))

          have df1: (2^(k−l))+((2^k))< ((2::nat)^(2*k−l))+((2^k))
            using ‹l < k› by (simp add: algebra-simps)

          have df2: ((2::nat)^(2*k−l))+((2^k)) < ((2::nat)^(2*k−l))*f
          proof−
            have k >1 using assms by simp
            have df: ((2::nat)^(k−l))+(1::nat) < ((2::nat)^(k−l))*f
            proof−
              obtain x::nat where xx: x=(2::nat)^(k−l) by simp
              have xxx: x ≥( 2::nat) using assms xx
                by (metis One-nat-def Suc-leI one-le-numeral power-increasing
                    semiring-normalization-rules(33) zero-less-diff)

              have c: x*f ≥ x*(2::nat) using aa by simp
              have x+(1::nat) < x*(2::nat)
                using auxiliary-ineq  xxx by linarith
              then have c2: ((2::nat)^(k−l)) + 1 < (2^(k−l))*2
                using xx by blast
              show ?thesis using c2 c xx
                by (metis diff-is-0-eq' le-trans nat-less-le zero-less-diff)
            qed

            show ?thesis using df aa assms
                      by (smt (verit, del-insts) JJ add.commute mult.commute
mult.left-commute mult-Suc-right mult-less-mono2 plus-1-eq-Suc zero-less-numeral
zero-less-power)
          qed
          show ?thesis using A-2 df1 df2 by linarith
        qed

        show ?thesis using assms A-1
          using diff-diff-left zero-less-diff by presburger
      qed

     show ?thesis using A aa assms
         by (smt (verit, ccfv-threshold) JJ a c d diff-mult-distrib mult.assoc
```

```
mult.commute mult-less-cancel2)
        qed

       have b3: ((2^(2*k−l)*(f^2))−((2^k)*f)−(2^(k−l)*f)+(2^(2*k−l))*f^2) =
                  (2*(2^(2*k−l)*(f^2))−((2^k)*f)−(2^(k−l)*f))
          using aaa minus-eq-nat-subst-order by auto

        show ?thesis using f1 by (metis b3 e mult-2)

      qed
      show ?thesis using f2 by simp
    qed

   have W8: 2^k*((2^(2*k−l+1)*(f^2))−((2^k)*f)−(2^(k−l)*f)) = (2^(3*k+1−l))*f^2−(2^(2*k−l))*f^2

    proof−
  have a: 2^k*(2^(2*k−l+1)*f^2−2^k*f−2^(k−l)*f) = 2^k*(2^(2*k−l+1)*f^2)−2^k*(2^k*f)−2^k*(2^(k-
        by (simp add: algebra-simps)

      have b: 2^k*(2^(2*k−l+1)*f^2)−2^k*(2^k*f)−2^k*(2^(k−l)*f) =
2^k*(2^(2*k−l+1)*f^2)−2^k*(2^k*f)−2^k*(2^(k−l)*f)
        by (simp add: algebra-simps)

      have c: 2^k*(2^(2*k−l+1)*f^2)−2^k*(2^k*f)−2^k*(2^(k−l)*f) =
2^(2*k+1−l+k)*f^2−2^k*(2^k*f)−2^k*(2^(k−l)*f)
        apply (simp add: algebra-simps power-add)
        using Suc-diff-le assms(1) by fastforce

      have d: 2^k*(2^(2*k−l+1)*(f^2))= (2^(3*k+1−l))*f^2
      by (smt (verit, ccfv-threshold) J JJ Nat.add-diff-assoc assms(1) less-imp-le-nat
mult.assoc mult.commute power-add)

      have e: 2^k*((2^(2*k−l+1)*(f^2))−((2^k)*f)−(2^(k−l)*f)) =
(2^(3*k+1−l))*f^2−(2^k)*((2^k)*f)−(2^k)*(2^(k−l)*f)

        using a b c d  One-nat-def  one-le-mult-iff
          Nat.add-diff-assoc assms(1) less-imp-le-nat by metis

      have ee: 2^k*((2^(2*k−l+1)*(f^2))−((2^k)*f)−((2::nat)^(k−l)*f))
= (2^(3*k+1−l))*f^2−( 2^k)*((2^k)*f)−(2^(2*k−l)*f)
        by (smt (verit) JJ a d mult.assoc mult.commute)

      have eee :
      −(( 2::nat)^(2*k−l))*(f^(2::nat)) =(−(( 2::nat)^(2*k))*f−(( 2::nat)^(2*k−l))*f)
        by (smt (verit, ccfv-SIG) auxicalc mult-minus-left of-nat-add of-nat-mult)

     have e4: 2^k*((2^(2*k−l+1)*(f^2))−((2^k)*f)−(2^(k−l)*f))=(2^(3*k+1−l))*f^2−(2^(2*k−l))*(f^2)

      proof−
```

```
    define A where A: A = 2^k*((2^(2*k−l+1)*(f^2))−((2^k)*f)−(2^(k−l)*f))
      define B where B: B = (2^(3*k+(1::nat)−l))*f^2
      define C where C: C = (2^k)*((2^k)*f)
      define D where D: D = (2^(2*k−l)*f)
      define E where E: E = (2^(2*k−l))*(f^2)
      have wq: A = B−C−D using ee A B C D by simp
      have wq1: −E = −C−D using eee C D E
        by (simp add: semiring-normalization-rules(36))
      have wq2: A = B−E using wq wq1 minus-eq-nat-subst by blast
      show ?thesis using wq2 A B E
        by metis
    qed
   show ?thesis using e4 by simp
  qed

  have Y: 2^k*p*q+2^k*r = (2^(3*k+1−l))*f^2−(2^(2*k−l))*f^2
    using W1 W2 W3 W4 W5 W6 W7 W8 by linarith
  show ?thesis using Y YY auxicalc by simp
 qed

 show ?thesis using Z Z1 Amicable-pair-equiv-def-conv assms One-nat-def one-le-mult-iff
      one-le-numeral less-imp-le-nat one-le-power
  by (metis prime-ge-1-nat)
qed
```

Another approach by Euler [2]:

```
theorem Euler-Rule-Amicable-1:
 fixes m n a :: nat
 assumes m ≥ 1 and n ≥ 1 and a ≥ 1
   and Esigma m = Esigma n and Esigma a * Esigma m = a*(m+n)
   and gcd a m =1 and gcd a n =1
   shows (a*m) Amic (a*n)

proof−
 have a: Esigma (a*m) =(Esigma a)*(Esigma m)
  using assms gcd-Esigma-mult by (simp add: mult.commute)

 have b: Esigma (a*m) = Esigma (a*n)

 proof−
  have c: Esigma (a*n) = (Esigma a)*(Esigma n)
   using gcd-Esigma-mult ‹gcd a n =1›
   by (metis assms(4) a )
  show ?thesis using c a assms by simp
   qed

   have d: Esigma (a*m) = a*m + a*n
  using a assms by (simp add: add-mult-distrib2)
 show ?thesis using a b d Amicable-pair-equiv-def-conv assms by (simp add:
```

Suc-leI)
qed

4 Thābit ibn Qurra's Rule and more examples

Euler's Rule (theorem Euler_Rule_Amicable) is actually a generalisation of the following rule by Thābit ibn Qurra from the 9th century [2]. Thābit ibn Qurra's Rule is the special case for $l = 1$ thus $f = 3$.

```
corollary Thabit-ibn-Qurra-Rule-Amicable:
  fixes k l f p q r :: nat
  assumes k > 1 and prime p and prime q  and prime r
  and p = 2^(k−1) ∗ 3 − 1 and q = 2^k ∗ 3 − 1 and r = 2^(2∗k−1) ∗ 9 − 1
  shows ((2^k)∗p∗q) Amic ((2^k)∗r)

proof−
   obtain l where l:l = (1::nat) by simp
   obtain f where f:f = (3::nat) by simp
   have k >l using l assms by simp
have  f =2^1+1 using f by simp
have  r =(2^(2∗k−1))∗(3^2)−1  using assms by simp
  show ?thesis  using assms Euler-Rule-Amicable  ‹f =2^1 +1›
     ‹ r =(2^(2∗k −1))∗(3^2) −1› l f
    by (metis le-numeral-extra(4))
qed
```

In the following three example of amicable pairs, instead of evaluating the sum of the divisors or using the properties of Euler's sigma function as it was done in the previous examples, we prove amicability more directly as we can apply Thābit ibn Qurra's Rule.

The following is the first example of an amicable pair known to the Pythagoreans and can be derived from Thābit ibn Qurra's Rule with $k = 2$ [2].

```
lemma Amicable-Example-Pythagoras:
  shows 220 Amic 284

proof−
  have a: (2::nat)>1 by simp
  have b: prime((3::nat)∗(2^(2−1))−1) by simp
  have c: prime((3::nat)∗(2^2)−1) by simp
  have d: prime((9::nat)∗(2^(2∗2−1))−1) by simp
 have e: ((2^2)∗(3∗(2^(2−1))−1)∗(3∗(2^2)−1))Amic((2^2)∗(9∗(2^(2∗2−1))−1))
    using Thabit-ibn-Qurra-Rule-Amicable a b c d
    by (metis mult.commute)

  have f: ((2::nat)^2)∗5∗11 = 220 by simp
  have g: ((2::nat)^2)∗71 = 284 by simp
    show ?thesis using  e f g by simp
```

qed

The following example of an amicable pair was (re)discovered by Fermat and can be derived from Thābit ibn Qurra's Rule with $k = 4$ [2].

lemma *Amicable-Example-Fermat*:
shows *17296 Amic 18416*

proof –
have *a*: $(4::nat)>1$ **by** *simp*
have *b*: $prime((3::nat)*(2^(4-1))-1)$ **by** *simp*
have *c*: $prime((3::nat)*(2^4)-1)$ **by** *simp*
have *d*: $prime\ (1151::nat)$ **by** (*pratt* (*code*))
have *e*: $(1151::nat) = 9*(2^(2*4-1))-1$ **by** *simp*
have *f*: $prime((9::nat)*(2^(2*4-1))-1)$ **using** *d e* **by** *metis*
have *g*: $((2^4)*(3*(2^(4-1))-1)*(3*(2^4)-1))\ Amic((2^4)*(9*(2^(2*4-1))-1))$
using *Thabit-ibn-Qurra-Rule-Amicable a b c f* **by** (*metis mult.commute*)
have *h*: $((2::nat)^4)*23*47 = 17296$ **by** *simp*
have *i*: $(((2::nat)^4)*1151) = 18416$ **by** *simp*

show *?thesis* **using** *g h i* **by** *simp*
qed

The following example of an amicable pair was (re)discovered by Descartes and can be derived from Thābit ibn Qurra's Rule with $k = 7$ [2].

lemma *Amicable-Example-Descartes*:
shows *9363584 Amic 9437056*

proof –
have *a*: $(7::nat)>1$ **by** *simp*
have *b*: $prime\ (191::nat)$ **by** (*pratt* (*code*))
have *c*: $((3::nat)*\ (2^(7-1))-1) =191$ **by** *simp*
have *d*: $prime((3::nat)*\ (2^(7-1))-1)$ **using** *b c* **by** *metis*
have *e*: $prime\ (383::nat)$ **by** (*pratt* (*code*))
have *f*: $(3::nat)*(2^7)-1 = 383$ **by** *simp*
have *g*: $prime\ ((3::nat)*(2^7)-1)$ **using** *e f* **by** *metis*
have *h*: $prime\ (73727::nat)$ **by** (*pratt* (*code*))
have *i*: $(9::nat)*(2^(2*7-1))-1 = 73727$ **by** *simp*
have *j*: $prime\ ((9::nat)*(2^(2*7-1))-1)$ **using** *i h* **by** *metis*
have *k*: $((2^7)*(3*(2^(7-1))-1)*(3*(2^7)-1))Amic((2^7)*(9*(2^(2*7-1))-1))$
using *Thabit-ibn-Qurra-Rule-Amicable a d g j* **by** (*metis mult.commute*)
have *l*: $((2::nat)^7)*\ 191*\ 383 = 9363584$ **by** *simp*
have *m*: $(((2::nat)^7)*\ 73727) = 9437056$ **by** *simp*

show *?thesis* **using** *a k l* **by** *simp*
qed

In fact, the Amicable Pair (220, 284) is Regular and of type (2,1):

lemma *regularAmicPairExample*: *regularAmicPair 220 284* $\wedge$ *typeAmic 220 284* $= [2,\ 1]$

```
proof−
  have a: 220 Amic 284 using Amicable-Example-Pythagoras by simp
  have b: gcd (220::nat) (284::nat) = 4  by eval
  have c: (220::nat) = 55∗4  by simp
  have d: (284::nat) = 71∗4  by simp
  have e: squarefree (55::nat) using squarefree-def by eval
  have f: squarefree (71::nat) using squarefree-def by eval
  have g: gcd (4::nat) (55::nat) =1 by eval
  have h: gcd (4::nat) (71::nat) =1 by eval

  have A: regularAmicPair 220 284
    by (simp add: a b e g f h gcd.commute regularAmicPair-def)
  have B: (card {i.∃ N. ( 220::nat) = N∗(4::nat) ∧ prime i ∧ i dvd N ∧ ¬ i dvd
4}) = 2

  proof−
    obtain N::nat where N: (220::nat) = N∗ 4
      by (metis  c)
    have NN:N=55 using N by simp
    have K1: prime(5::nat) by simp
    have K2: prime(11::nat) by simp
    have KK2:  ¬ prime (55::nat) by simp
    have KK3:  ¬ prime (1::nat) by simp
    have K: set(divisors-nat 55 ) = {1, 5, 11, 55} by eval
    have KK: {i. i dvd (55::nat)} = {1, 5, 11, 55}
      using K divisors-nat  divisors-nat-def by auto
    have K3 : ¬ (5::nat) dvd 4 by simp
    have K4 : ¬ (11::nat) dvd 4 by simp
    have K55: (1::nat) ∉ {i. prime i ∧ i dvd 55} using KK3 by simp
    have K56: (55::nat) ∉ {i. prime i ∧ i dvd 55} using KK2 by simp
    have K57: (5::nat) ∈ {i. prime i ∧ i dvd 55} using K1 by simp
    have K58: (11::nat) ∈ {i. prime i ∧ i dvd 55} using K2 by simp
    have K5:  {i.( prime i ∧ i dvd (55::nat) ∧ ¬ i dvd 4)} = {5, 11}

    proof−
      have K66: {i.(prime i ∧ i dvd (55::nat) ∧ ¬ i dvd 4)}=
{i. prime i} ∩ {i. i dvd 55} ∩ { i. ¬ i dvd 4}
        by blast
      show ?thesis using K66  K K1 K2 KK2 KK3  K3 K4 KK K55 K56 K57 K58
divisors-nat-def
          divisors-nat by auto
    qed
    have K6:  card ({(5::nat), (11::nat)}) = 2  by simp
    show ?thesis using K5 K6 by simp
  qed

  have C: (card {i. ∃N. (284::nat) = N∗4 ∧ prime i ∧ i dvd N ∧ ¬ i dvd 4} ) =
1
  proof−
```

```
    obtain N::nat where N: 284 = N*4
      by (metis d)
    have NN: N= 71 using N by simp
    have K: set(divisors-nat 71 ) = {1, 71 } by eval
    have KK: {i. i dvd (71::nat)} = {1, 71}
      using K divisors-nat  divisors-nat-def by auto

    have K55:(1::nat) ∉ {i. prime i ∧ i dvd 71} by simp
    have K58: (71::nat) ∈ {i. prime i ∧ i dvd 71} by simp
    have K5: {i. prime i ∧ i dvd 71 ∧ ¬ i dvd 4} = {(71::nat)}
    proof−
      have K66: {i. prime i ∧ i dvd 71 ∧ ¬ i dvd 4}=
{i. prime i} ∩ {i. i dvd 71} ∩ { i. ¬ i dvd 4}
        by blast
      show ?thesis using  K KK K55 K58
        by (auto simp add: divisors-nat-def K66 divisors-nat)
    qed
    have K6: card ({(71::nat)}) = 1  by simp
    show ?thesis using K5 K6 by simp
  qed

  show ?thesis using A B C
    by (simp add: typeAmic-def b)
qed

lemma abundant220ex: abundant-number 220
proof−
  have 220 Amic 284 using Amicable-Example-Pythagoras by simp
  moreover have (220::nat) < 284 by simp
  ultimately show ?thesis using Amicable-pair-abundant Amicable-pair-sym
    by blast
qed

lemma deficient284ex: deficient-number 284
proof−
  have 220 Amic 284 using Amicable-Example-Pythagoras by simp
  moreover have (220::nat) < 284 by simp
  ultimately show ?thesis using Amicable-pair-deficient Amicable-pair-sym
    by blast
qed
```

5 Te Riele's Rule and Borho's Rule with breeders

With the following rule [2] we can get an amicable pair from a known amicable pair under certain conditions.

```
theorem teRiele-Rule-Amicable:
  fixes a u p r c q :: nat
  assumes a ≥ 1 and u ≥ 1
```

```
    and prime p and prime r and prime c and prime q and r ≠ c
    and ¬(p dvd a) and (a∗u) Amic (a∗p) and gcd a (r∗c)=1
    and q = r+c+u and gcd (a∗u) q =1 and r∗c = p∗(r +c+ u) + p+u
  shows (a∗u∗q) Amic (a∗r∗c)

proof−
  have p+1 >0 using assms by simp
  have Z1:  r∗c = p∗q+p+u using assms by auto
  have Z2: (r+1)∗(c+1) = (q+1)∗(p+1)
  proof−
    have y: (q+1)∗(p+1) = q∗p + q+ p+1  by simp
    have yy: (r+1)∗(c+1) = r∗c + r+ c+1 by simp
    show ?thesis using assms y Z1 yy  by simp
  qed

  have ∗: Esigma(a) = (a∗(u+p)/(p+1))
  proof−
    have d: Esigma (a∗p) = (Esigma a)∗(Esigma p)
      using assms gcd-Esigma-mult ‹prime p› ‹¬ (p dvd a)›
      by (metis gcd-unique-nat prime-nat-iff)
    have dd : Esigma (a∗p) =(Esigma a)∗(p+1)
      using d assms  prime-sum-div by simp
    have ddd: Esigma (a∗p) = a∗(u+p)
      using assms unfolding Amicable-pair-def
       by (metis Esigma-properdiv-set One-nat-def add-mult-distrib2 one-le-mult-iff
prime-ge-1-nat)
    show ?thesis using d dd ddd Esigmanotzero assms(3) dvd-triv-right
        nonzero-mult-div-cancel-right prime-nat-iff prime-sum-div real-of-nat-div
      by (metis ‹0 < p + 1› neq0-conv)
  qed

  have Esigma(r) = (r+1) using assms prime-sum-div by blast
  have Esigma(c) = (c+1) using assms prime-sum-div by blast
  have Esigma (a∗r∗c) = (Esigma a)∗(Esigma r)∗(Esigma c)
  proof−
    have h: Esigma (a∗r∗c) = (Esigma a)∗(Esigma (r∗c))
      using assms gcd-Esigma-mult
      by (metis mult.assoc mult.commute)
   have hh:  Esigma (r∗c) = (Esigma r)∗(Esigma c) using assms prime-Esigma-mult
      by (metis semiring-normalization-rules(7))

    show ?thesis using h hh by auto
  qed

  have A: Esigma (a∗u∗q) = Esigma (a∗r∗c)
  proof−
    have wk: Esigma (a∗u∗q) = Esigma (a∗u)∗(q+1)
      using assms gcd-Esigma-mult  by (simp add: prime-sum-div)
   have wk1: Esigma (a∗u) = a∗(u+p) using assms unfolding Amicable-pair-equiv-def
```

by (*metis Amicable-pair-def Esigma-properdiv-set Suc-eq-plus1 add.commute add-0 add-mult-distrib2 one-le-mult-iff*)

have *w3*: *Esigma* $(a*u*q) = a*(u+p)*(q+1)$ **using** *wk wk1* **by** *simp*
have *w4*: *Esigma* $(a*r*c) =(Esigma\ a)*(r+1) * (c+1)$ **using** *assms*
by (*simp add*: ‹*Esigma* $(a*r*c) = Esigma\ a * Esigma\ r * Esigma\ c$› ‹*Esigma* $c = c + 1$›
‹*Esigma* $r = r+1$›)

have *we*: $a*(u+p)*(q+1) = (Esigma\ a)*(r+1)*(c+1)$
proof−
have $(Esigma\ a)*(r+1)*(c+1) = (a*(u+p)/(p+1))*(r+1)*(c+1)$
by (*metis* ‹*real* (*Esigma a*) = *real* $(a*(u+p))/real(p+1)$› *of-nat-mult*)
then have $(Esigma\ a)*(r+1)*(c+1) = (a*(u+p)/(p+1))*(q+1)*(p+1)$ **using** *Z2*
by (*metis of-nat-mult semiring-normalization-rules*(*18*))
then show *?thesis* **using** *assms*
apply (*simp add*: *divide-simps*)
by (*metis* (*no-types*, *opaque-lifting*) *mult.commute mult-Suc of-nat-Suc of-nat-add of-nat-eq-iff of-nat-mult*)
qed

show *?thesis* **using** *we w3 w4* **by** *simp*
qed

have *B* : *Esigma* $(a*r*c) = (a*u*q)+(a*r*c)$
proof−
have *a1*: $(u+p)*(q+1) = (u*q+p*q+p+u)$ **using** *assms add-mult-distrib* **by** *auto*
have *a2*: $(u+p)*(q+1)*(p+1) = (u*q+p*q+p+u)*(p+1)$ **using** *a1 assms* **by** *metis*
have *a3*: $(u+p)*(r+1)*(c+1) = (u*q+p*q+p+u)*(p+1)$ **using** *assms a2 Z2*
by (*metis semiring-normalization-rules*(*18*))
have *a4*: $a*(u+p)* (r+1)*(c+1) = a*(u*q+ p*q+p+u)*(p+1)$ **using** *assms a3*
by (*metis semiring-normalization-rules*(*18*))
have *a5*: $a*(u+p)*(r+1)*(c+1) = a*(u*q+r*c)*(p+1)$ **using** *assms a4 Z1*
by (*simp add*: *semiring-normalization-rules*(*21*))
have *a6*: $(a*(u+p)*(r+1)*(c+1))/(p+1) =(a*(u*q+ r*c)* (p+1))/(p+1)$ **using** *assms a5*
semiring-normalization-rules(*21*) ‹$p+1 >0$› **by** *auto*
have *a7*: $(a*(u+p)*(r+1)*(c+1))/(p+1) =(a*(u*q+ r*c))$ **using** *assms a6* ‹$p+1 >0$›
by (*metis neq0-conv nonzero-mult-div-cancel-right of-nat-eq-0-iff of-nat-mult*)
have *a8*:$(a*(u+p)/(p+1))*(r+1)*(c+1) = a*(u*q+r*c)$ **using** *assms a7* ‹$p+1 >0$›
by (*metis of-nat-mult times-divide-eq-left*)
have *a9*: $(Esigma\ a)* Esigma(r)* Esigma(c) = a*(u*q+ r*c)$ **using** *a8 assms*
‹ $Esigma(r) = (r+1)$› ‹ $Esigma(c) = (c+1)$›

```
      by (metis ‹real (Esigma a) = real (a∗(u + p))/real(p + 1)› of-nat-eq-iff
of-nat-mult)
    have a10: Esigma(a∗r∗c) = a∗(u∗q+ r∗c) using a9 assms
      ‹Esigma (a∗r∗c) = (Esigma a)∗(Esigma r)∗(Esigma c)› by simp

    show ?thesis using a10 assms
      by (simp add: add-mult-distrib2 mult.assoc)
  qed

  show ?thesis
   using assms by (metis A B One-nat-def one-le-mult-iff Amicable-pair-equiv-def-conv
prime-ge-1-nat)
 qed
```

By replacing the assumption that $(a*u)$ *Amic* $(a*p)$ in the above rule by te Riele with the assumption that $(a*u)$ *breeder* u, we obtain Borho's Rule with breeders [2].

```
theorem Borho-Rule-breeders-Amicable:
  fixes a u r c q x :: nat
  assumes x ≥ 1 and a ≥ 1 and u ≥ 1
      and prime r and prime c and  prime q and r ≠ c
     and Esigma (a∗u) = a∗u + a∗x Esigma (a∗u) = (Esigma a)∗(x+1) and gcd
a (r ∗ c) =1
      and gcd (a∗u) q = 1 and r ∗ c = x+u + x∗u +r∗x +x∗c and q = r+c+u
  shows (a∗u∗q) Amic (a∗r∗c)

proof−
  have a: Esigma(a∗u∗q) = Esigma(a∗u)∗Esigma(q)
    using assms gcd-Esigma-mult by simp
  have a1: Esigma(a∗r∗c) = (Esigma a)∗Esigma(r∗c)
    using assms gcd-Esigma-mult by (metis mult.assoc mult.commute)
  have a2: Esigma(a∗r∗c) = (Esigma a)∗(r+1)∗(c+1)
    using a1 assms
   by (metis mult.commute mult.left-commute prime-Esigma-mult prime-sum-div)

  have A: Esigma (a∗u∗q) = Esigma(a∗r∗c)
  proof−
    have d: Esigma(a)∗(r+1)∗(c+1) = Esigma(a∗u)∗(q+1)
    proof−
      have d1: (r+1)∗(c+1) =(x+1)∗(q+1)
      proof−
        have ce: (r+1)∗(c+1) = r∗c+r+c+1 by simp
        have ce1: (r+1)∗(c+1) = x+u+x∗u+r∗x+x∗c+r+c+1
          using ce assms by simp
        have de: (x+1)∗(q+1) = x∗q +1+x+q by simp
        have de1: (x+1)∗(q+1) = x∗(r+c+u)+1+x+ r+c+u
          using assms de by simp
        show ?thesis using de1 ce1 add-mult-distrib2 by auto
      qed
```

```
      show ?thesis using d1 assms
        by (metis semiring-normalization-rules(18))
    qed

    show ?thesis using d a2
      by (simp add: a assms(6) prime-sum-div)
  qed

  have B: Esigma (a*u*q) = a*u*q + a*r*c
  proof−
    have Esigma (a*u*q) = Esigma(a*u)*(q+1)
      using a assms
      by (simp add: prime-sum-div)
    with assms have Esigma (a*u*q) = (a*u+ a*x)*(q+1)
      by auto
     with assms have Esigma (a*u*q) = a*u*q +a*u+ a*x*q+ a*x
       using add-mult-distrib by simp
    with assms show ?thesis
      by (simp add: distrib-left mult.commute mult.left-commute)
  qed

  show ?thesis using A B assms
    using Amicable-pair-equiv-def-conv prime-ge-1-nat by force
qed

no-notation divisor (infixr divisor 80)
```

6 Acknowledgements

The author was supported by the ERC Advanced Grant ALEXANDRIA (Project 742178) funded by the European Research Council and led by Professor Lawrence Paulson at the University of Cambridge, UK. Many thanks to Lawrence Paulson for his help and suggestions. Number divisors were initially looked up on https://onlinemathtools.com/find-all-divisors.

end

References

[1] E. Escott. Amicable numbers. *Scripta Mathematica*, 12:61–72, 1946.

[2] M. García, J. Pedersen, and H. te Riele. Amicable pairs, a survey. *REPORT MAS-R0307*, 2003.

[3] M. García, J. Pedersen, and H. te Riele. Amicable pairs, a survey. *Fields Institute Communications*, 41:1–19, 2004.

[4] E. Sandifer. Amicable pairs. 2005. How Euler Did It, The Euler Archive.

[5] Wikipedia. Aliquot sum. https://en.wikipedia.org/wiki/Aliquot_sum, 2020.

[6] Wikipedia. Amicable numbers. https://en.wikipedia.org/wiki/Amicable_numbers, 2020.

[7] Wikipedia. Betrothed numbers. https://en.wikipedia.org/wiki/Betrothed_numbers, 2020.