Exponents 3 and 4 of Fermat's Last Theorem and the Parametrisation of Pythagorean Triples

Roelof Oosterhuis
University of Groningen

May 26, 2024

Abstract

This document gives a formal proof of the cases $n = 3$ and $n = 4$ (and all their multiples) of Fermat's Last Theorem: if $n > 2$ then for all integers x, y, z:

$$x^n + y^n = z^n \Longrightarrow xyz = 0.$$

Both proofs only use facts about the integers and are developed along the lines of the standard proofs (see, for example, sections 1 and 2 of the book by Edwards [Edw77]).

First, the framework of 'infinite descent' is being formalised and in both proofs there is a central role for the lemma

$$coprime ab \ \wedge \ ab = c^n \Longrightarrow \exists\ k : |a| = k^n.$$

Furthermore, the proof of the case $n = 4$ uses a parametrisation of the Pythagorean triples. The proof of the case $n = 3$ contains a study of the quadratic form $x^2 + 3y^2$. This study is completed with a result on which prime numbers can be written as $x^2 + 3y^2$.

The case $n = 4$ of FLT, in contrast to the case $n = 3$, has already been formalised (in the proof assistant Coq) [DM05]. The parametrisation of the Pythagorean Triples can be found as number 23 on the list of 'top 100 mathematical theorems' [Wie].

This research is part of an M.Sc. thesis under supervision of Jaap Top and Wim H. Hesselink (RU Groningen). The author wants to thank Clemens Ballarin (TU München) and Freek Wiedijk (RU Nijmegen) for their support. For more information see [Oos07].

Contents

1 Pythagorean triples and Fermat's last theorem, case $n = 4$

```
theory Fermat4
imports HOL−Computational-Algebra.Primes
begin

context
begin


private lemma nat-relprime-power-divisors:
  assumes n0: 0 < n and abc: (a::nat)∗b = c^n and relprime: coprime a b
  shows ∃ k. a = k^n
using assms proof (induct c arbitrary: a b rule: nat-less-induct)
case (1 c)
  show ?case
  proof (cases a > 1)
  case False
    hence a = 0 ∨ a = 1 by linarith
    thus ?thesis using n0 power-one zero-power by (simp only: eq-sym-conv) blast
  next
  case True
    then obtain p where p: prime p p dvd a using prime-factor-nat[of a] by blast
    hence h1: p dvd (c^n) using 1(3) dvd-mult2[of p a b] by presburger
    hence (p^n) dvd (c^n)
      using p(1) prime-dvd-power-nat[of p c n] dvd-power-same[of p c n] by blast
    moreover have h2: ¬ p dvd b
      using p ‹coprime a b› coprime-common-divisor-nat [of a b p] by auto
    hence ¬ (p^n) dvd b using n0 p(1)
      by (auto intro: dvd-trans dvd-power[of n p])
    ultimately have (p^n) dvd a
      using 1.prems p(1) prime-elem-divprod-pow [of p a b n] by simp
    then obtain a′ c′ where ac: a = p^n ∗ a′ c = p ∗ c′
     using h1 dvdE[of p^n a] dvdE[of p c] prime-dvd-power-nat[of p c n] p(1) by meson
    hence p^n ∗ (a′ ∗ b) = p^n ∗ c′^n using 1(3)
      by (simp add: power-mult-distrib semiring-normalization-rules(18))
    hence a′ ∗ b = c′^n using p(1) by auto
    moreover have coprime a′ b using 1(4) ac(1)
      by (simp add: ac-simps)
    moreover have 0 < b 0 < a using h2 dvd-0-right gr0I True by fastforce+
    then have 0 < c 1 < p
      using p ‹a ∗ b = c ^ n› n0 nat-0-less-mult-iff [of a b] n0
      by (auto simp add: prime-gt-Suc-0-nat)
    hence c′ < c using ac(2) by simp
    ultimately obtain k where a′ = k^n using 1(1) n0 by presburger
    hence a = (p∗k)^n using ac(1) by (simp add: power-mult-distrib)
    thus ?thesis by blast
  qed
qed
```

private lemma int-relprime-power-divisors:
  assumes 0 < n **and** 0 ≤ a **and** 0 ≤ b **and** (a::int) * b = c ^ n **and** coprime a b
  shows ∃ k. a = k^n
proof (cases a = 0)
  case False
  from ‹0 ≤ a› ‹0 ≤ b› ‹a * b = c ^ n›[symmetric] **have** 0 ≤ c ^ n
    by simp
  hence c^n = |c|^n **using** power-even-abs[of n c] zero-le-power-eq[of c n] **by** linarith
  hence a * b = |c|^n **using** assms(4) **by** presburger
  hence nat a * nat b = (nat |c|)^n **using** nat-mult-distrib[of a b] assms(2)
    by (simp add: nat-power-eq)
  moreover have 0 ≤ b **using** assms mult-less-0-iff[of a b] False **by** auto
  with ‹0 ≤ a› ‹coprime a b› **have** coprime (nat a) (nat b)
    using coprime-nat-abs-left-iff [of a nat b] **by** simp
  ultimately have ∃ k. nat a = k^n
    using nat-relprime-power-divisors[of n nat a nat b nat |c|] assms(1) **by** blast
  thus ?thesis **using** assms(2) int-nat-eq[of a] **by** fastforce
qed (simp add: zero-power[of n] assms(1))

Proof of Fermat's last theorem for the case $n = 4$:

$$\forall x, y, z:\ x^4 + y^4 = z^4 \Longrightarrow xyz = 0.$$

private lemma nat-power2-diff: a ≥ (b::nat) ⟹ (a−b)^2 = a^2 + b^2 − 2*a*b
proof −
  assume a-ge-b: a ≥ b
  hence a2-ge-b2: a^2 ≥ b^2 **by** (simp only: power-mono)
  from a-ge-b **have** ab-ge-b2: a*b ≥ b^2 **by** (simp add: power2-eq-square)
  have b*(a−b) + (a−b)^2 = a*(a−b) **by** (simp add: power2-eq-square diff-mult-distrib)
  also have . . . = a*b + a^2 + (b^2 − b^2) − 2*a*b
    by (simp add: diff-mult-distrib2 power2-eq-square)
  also with a2-ge-b2 **have** . . . = a*b + (a^2 − b^2) + b^2 − 2*a*b
    by (simp add: power2-eq-square)
  also with ab-ge-b2 **have** . . . = (a*b − b^2) + a^2 + b^2 − 2*a*b **by** auto
  also have . . . = b*(a−b) + a^2 + b^2 − 2*a*b
    by (simp only: diff-mult-distrib2 power2-eq-square mult.commute)
  finally show ?thesis **by** arith
qed

private lemma nat-power-le-imp-le-base: ⟦ n ≠ 0; a^n ≤ b^n ⟧ ⟹ (a::nat) ≤ b
  by simp

private lemma nat-power-inject-base: ⟦ n ≠ 0; a^n = b^n ⟧ ⟹ (a::nat) = b
proof −
  assume n ≠ 0 **and** ab: a^n = b^n
  then obtain m **where** n = Suc m **by** (frule-tac n=n **in** not0-implies-Suc, auto)
  with ab **have** a^Suc m = b^Suc m **and** a ≥ 0 **and** b ≥ 0 **by** auto
  thus ?thesis **by** (rule power-inject-base)
qed

1.1 Parametrisation of Pythagorean triples (over ℕ and ℤ)

private theorem nat-euclid-pyth-triples:

```
  assumes abc: (a::nat)^2 + b^2 = c^2 and ab-relprime: coprime a b and aodd: odd a
  shows ∃ p q. a = p^2 − q^2 ∧ b = 2*p*q ∧ c = p^2 + q^2 ∧ coprime p q
proof −
  have two0: (2::nat) ≠ 0 by simp
  from abc have a2cb: a^2 = c^2 − b^2 by arith
  — factor a² in coprime factors (c − b) and (c + b); hence both are squares
  have a2factor: a^2 = (c−b)*(c+b)
  proof −
    have c*b − c*b = 0 by simp
    with a2cb have a^2 = c*c + c*b − c*b − b*b by (simp add: power2-eq-square)
    also have ... = c*(c+b) − b*(c+b)
      by (simp add: add-mult-distrib2 add-mult-distrib mult.commute)
    finally show ?thesis by (simp only: diff-mult-distrib)
  qed
  have a-nonzero: a ≠ 0
  proof (rule ccontr)
    assume ¬ a ≠ 0 hence a = 0 by simp
    with aodd have odd (0::nat) by simp
    thus False by simp
  qed
  have b-less-c: b < c
  proof −
    from abc have b^2 ≤ c^2 by linarith
    with two0 have b ≤ c by (rule-tac n=2 in nat-power-le-imp-le-base)
    moreover haveb ≠ c
    proof
      assume b=c with a2cb have a^2 = 0 by simp
      with a-nonzero show False by (simp add: power2-eq-square)
    qed
    ultimately show ?thesis by auto
  qed
  hence b2-le-c2: b^2 ≤ c^2 by (simp add: power-mono)
  have bc-relprime: coprime b c
  proof −
    from b2-le-c2 have cancelb2: c^2−b^2+b^2 = c^2 by auto
    let ?g = gcd b c
    have ?g^2 = gcd (b^2) (c^2) by simp
    with cancelb2 have ?g^2 = gcd (b^2) (c^2−b^2+b^2) by simp
    hence ?g^2 = gcd (b^2) (c^2−b^2) using gcd-add2[of b^2 c^2 − b^2]
      by (simp add: algebra-simps del: gcd-add1)
    with a2cb have ?g^2 dvd a^2 by (simp only: gcd-dvd2)
    hence ?g dvd a ∧ ?g dvd b by simp
    hence ?g dvd gcd a b by (simp only: gcd-greatest)
    with ab-relprime show ?thesis
      by (simp add: ac-simps gcd-eq-1-imp-coprime)
  qed
  have p2: prime (2::nat) by simp
  have factors-odd: odd (c−b) ∧ odd (c+b)
  proof (auto simp only: ccontr)
    assume even (c−b)
    with a2factor have 2 dvd a^2 by (simp only: dvd-mult2)
    with p2 have 2 dvd a by auto
```

```
    with aodd show False by simp
  next
    assume even (c+b)
    with a2factor have 2 dvd a^2 by (simp only: dvd-mult)
    with p2 have 2 dvd a by auto
    with aodd show False by simp
  qed
  have cb1: c−b + (c+b) = 2∗c
  proof −
    have c−b + (c+b) = ((c−b)+b)+c by simp
    also with b-less-c have ... = (c+b−b)+c by (simp only: diff-add-assoc2)
    also have ... = c+c by simp
    finally show ?thesis by simp
  qed
  have cb2: 2∗b + (c−b) = c+b
  proof −
    have 2∗b + (c−b) = b+b + (c − b) by auto
    also have ... = b + ((c−b)+b) by simp
    also with b-less-c have  ... = b + (c+b−b) by (simp only: diff-add-assoc2)
    finally show ?thesis by simp
  qed
  have factors-relprime: coprime (c−b) (c+b)
  proof −
    let ?g = gcd (c−b) (c+b)
    have cb1: c−b + (c+b) = 2∗c
    proof −
      have c−b + (c+b) = ((c−b)+b)+c by simp
      also with b-less-c have ... = (c+b−b)+c by (simp only: diff-add-assoc2)
      also have ... = c+c by simp
      finally show ?thesis by simp
    qed
    have ?g = gcd (c−b + (c+b)) (c+b) by simp
    with cb1 have ?g = gcd (2∗c) (c+b) by (rule-tac a=c−b + (c+b) in back-subst)
    hence g2c: ?g dvd 2∗c by (simp only: gcd-dvd1)
    have gcd (c−b) (2∗b + (c−b)) = gcd (c−b) (2∗b)
      using gcd-add2[of c − b 2∗b + (c − b)] by (simp add: algebra-simps)
    with cb2 have ?g = gcd (c−b) (2∗b) by (rule-tac a=2∗b + (c−b) in back-subst)
    hence g2b: ?g dvd 2∗b by (simp only: gcd-dvd2)
    with g2c have ?g dvd 2 ∗ gcd b c by (simp only: gcd-greatest gcd-mult-distrib-nat)
    with bc-relprime have ?g dvd 2 by simp
    moreover have ?g ≠ 0
      using b-less-c by auto
    ultimately have 1 ≤ ?g ?g ≤ 2
      by (simp-all add: dvd-imp-le)
    then have g1or2: ?g = 2 ∨ ?g = 1
      by arith
    moreover have ?g ≠ 2
    proof
      assume ?g = 2
      moreover have ?g dvd c − b
        by simp
      ultimately show False
```

```
        using factors-odd by simp
    qed
    ultimately show ?thesis
      by (auto intro: gcd-eq-1-imp-coprime)
  qed
  from a2factor have (c−b)*(c+b) = a^2 and (2::nat) >1  by auto
  with factors-relprime have ∃ k. c−b = k^2
    by (simp only: nat-relprime-power-divisors)
  then obtain r where r: c−b = r^2 by auto
  from a2factor have (c+b)*(c−b) = a^2 and (2::nat) >1  by auto
  with factors-relprime have ∃ k. c+b = k^2
    by (simp only: nat-relprime-power-divisors ac-simps)
  then obtain s where s: c+b = s^2 by auto
  — now p := (s + r)/2 and q := (s − r)/2 is our solution
  have rs-odd: odd r ∧ odd s
  proof (auto dest: ccontr)
    assume even r hence 2 dvd rby presburger
    with r have  2 dvd (c−b) by (simp only: power2-eq-square dvd-mult)
    with factors-odd show False by auto
  next
    assume even s hence 2 dvd s by presburger
    with s have 2 dvd (c+b) by (simp only: power2-eq-square dvd-mult)
    with factors-odd show False by auto
  qed
  obtain m where m: m = s−r by simp
  from r s have r^2 ≤ s^2 by arith
  with two0 have r ≤ s by (rule-tac n=2 in nat-power-le-imp-le-base)
  with m have m2: s = r + m by simp
  have even m
  proof (rule ccontr)
    assume odd m with rs-odd and m2 show False by presburger
  qed
  then obtain q where m = 2*q ..
  with m2 have q: s = r + 2*q by simp
  obtain p where p: p = r+q by simp
  have c: c = p^2 + q^2
  proof −
    from cb1 and r and s have 2*c = r^2 + s^2 by simp
    also with q have ... = 2*r^2+(2*q)^2+2*r*(2*q) by algebra
    also have ... = 2*r^2+2^2*q^2+2*2*q*r by (simp add: power-mult-distrib)
    also have ... = 2*(r^2+2*q*r+q^2)+2*q^2 by (simp add: power2-eq-square)
    also with p have ... = 2*p^2+2*q^2 by algebra
    finally show ?thesis by auto
  qed
  moreover have b: b = 2*p*q
  proof −
    from cb2 and r and s have 2*b = s^2 − r^2 by arith
    also with q have ... = (2*q)^2 + 2*r*(2*q) by (simp add: power2-sum)
    also with p have ... = 4*q*p by (simp add: power2-eq-square add-mult-distrib2)
    finally show ?thesis by auto
  qed
  moreover have a: a = p^2 − q^2
```

```
  proof −
    from p have p≥q by simp
    hence p2-ge-q2: p^2 ≥ q^2 by (simp only: power-mono)
    from a2cb and b and c have a^2 = (p^2 + q^2)^2 − (2∗p∗q)^2 by simp
    also have . . . = (p^2)^2 + (q^2)^2 − 2∗(p^2)∗(q^2)
      by (auto simp add: power2-sum power-mult-distrib ac-simps)
    also with p2-ge-q2 have . . . = (p^2 − q^2)^2 by (simp only: nat-power2-diff)
    finally have a^2 = (p^2 − q^2)^2 by simp
    with two0 show ?thesis by (rule-tac n=2 in nat-power-inject-base)
  qed
  moreover have coprime p q
  proof −
    let ?k = gcd p q
    have ?k dvd p ∧ ?k dvd q by simp
    with b and a have ?k dvd a ∧ ?k dvd b
      by (simp add: power2-eq-square)
    hence ?k dvd gcd a b by (simp only: gcd-greatest)
    with ab-relprime show ?thesis
      by (auto intro: gcd-eq-1-imp-coprime)
  qed
  ultimately show ?thesis by auto
qed
```

Now for the case of integers. Based on *nat-euclid-pyth-triples*.

```
private corollary int-euclid-pyth-triples: ⟦ coprime (a::int) b; odd a; a^2 + b^2 = c^2
⟧
  ⟹ ∃ p q. a = p^2 − q^2 ∧ b = 2∗p∗q ∧ |c| = p^2 + q^2 ∧ coprime p q
proof −
  assume ab-rel: coprime a b and aodd: odd a and abc: a^2 + b^2 = c^2
  let ?a = nat|a|
  let ?b = nat|b|
  let ?c = nat|c|
  have ab2-pos: a^2 ≥ 0 ∧ b^2 ≥ 0 by simp
  hence nat(a^2) + nat(b^2) = nat(a^2 + b^2) by (simp only: nat-add-distrib)
  with abc have nat(a^2) + nat(b^2) = nat(c^2) by presburger
  hence nat(|a|^2) + nat(|b|^2) = nat(|c|^2) by simp
  hence new-abc: ?a^2 + ?b^2 = ?c^2
    by (simp only: nat-mult-distrib power2-eq-square nat-add-distrib)
  moreover from ab-rel have new-ab-rel: coprime ?a ?b
    by (simp add: gcd-int-def)
  moreover have new-a-odd: odd ?a using aodd
    by simp
  ultimately have
    ∃ p q. ?a = p^2−q^2 ∧ ?b = 2∗p∗q ∧ ?c = p^2 + q^2 ∧ coprime p q
    by (rule-tac a=?a and b = ?b and c=?c in nat-euclid-pyth-triples)
  then obtain m and n where mn:
    ?a = m^2−n^2 ∧ ?b = 2∗m∗n ∧ ?c = m^2 + n^2 ∧ coprime m n by auto
  have n^2 ≤ m^2
  proof (rule ccontr)
    assume ¬ n^2 ≤ m^2
    with mn have ?a = 0 by auto
    with new-a-odd show False by simp
```

```
  qed
  moreover from mn have int ?a = int(m^2 − n^2) and int ?b = int(2∗m∗n)
    and int ?c = int(m^2 + n^2) by auto
  ultimately have |a| = int(m^2) − int(n^2) and |b| = int(2∗m∗n)
    and |c| = int(m^2) + int(n^2) by (simp add: of-nat-diff)+
  hence absabc: |a| = (int m)^2 − (int n)^2 ∧ |b| = 2∗(int m)∗int n
    ∧ |c| = (int m)^2 + (int n)^2 by (simp add: power2-eq-square)
  from mn have mn-rel: coprime (int m) (int n)
    by (simp add: gcd-int-def)
  show ∃ p q. a = p^2 − q^2 ∧ b = 2∗p∗q ∧ |c| = p^2 + q^2 ∧ coprime p q
    (is ∃ p q. ?Q p q)
  proof (cases)
    assume apos: a ≥ 0 then obtain p where p: p = int m by simp
    hence ∃ q. ?Q p q
    proof (cases)
      assume bpos: b ≥ 0 then obtain q where q = int n by simp
      with p apos bpos absabc mn-rel have ?Q p q by simp
      thus ?thesis by (rule exI)
    next
      assume ¬ b≥0 hence bneg: b<0 by simp
      then obtain q where q = − int n by simp
      with p apos bneg absabc mn-rel have ?Q p q by simp
      thus ?thesis by (rule exI)
    qed
    thus ?thesis by (simp only: exI)
  next
    assume ¬ a≥0 hence aneg: a<0 by simp
    then obtain p where p: p = int n by simp
    hence ∃ q. ?Q p q
    proof (cases)
      assume bpos: b ≥ 0 then obtain q where q = int m by simp
      with p aneg bpos absabc mn-rel have ?Q p q
        by (simp add: ac-simps)
      thus ?thesis by (rule exI)
    next
      assume ¬ b≥0 hence bneg: b<0 by simp
      then obtain q where q = − int m by simp
      with p aneg bneg absabc mn-rel have ?Q p q
        by (simp add: ac-simps)
      thus ?thesis by (rule exI)
    qed
    thus ?thesis by (simp only: exI)
  qed
qed
```

1.2 Fermat's last theorem, case $n = 4$

Core of the proof. Constructs a smaller solution over $\mathbb{Z}$ of

$$a^4 + b^4 = c^2 \wedge coprime\, a\, b \wedge abc \neq 0 \wedge a \text{ odd}.$$

private lemma *smaller-fermat4*:

```
 assumes abc: (a::int)^4+b^4=c^2 and abc0: a*b*c ≠ 0 and aodd: odd a
   and ab-relprime: coprime a b
 shows
 ∃ p q r. (p^4+q^4=r^2 ∧ p*q*r ≠ 0 ∧ odd p ∧ coprime p q ∧ r^2 < c^2)
proof −
 — put equation in shape of a pythagorean triple and obtain u and v
 from ab-relprime have a2b2relprime: coprime (a^2) (b^2)
   by simp
 moreover from aodd have odd (a^2) by presburger
 moreover from abc have (a^2)^2 + (b^2)^2 = c^2 by simp
 ultimately obtain u and v where uvabc:
   a^2 = u^2−v^2 ∧ b^2 = 2*u*v ∧ |c| = u^2 + v^2 ∧ coprime u v
   by (frule-tac a=a^2 in int-euclid-pyth-triples, auto)
 with abc0 have uv0: u≠0 ∧ v≠0 by auto
 have av-relprime: coprime a v
 proof −
   have gcd a v dvd gcd (a^2) v by (simp add: power2-eq-square)
   moreover from uvabc have gcd v (a^2) dvd gcd (b^2) (a^2)
     by simp
   with a2b2relprime have gcd (a^2) v dvd (1::int)
     by (simp add: ac-simps)
   ultimately have gcd a v dvd 1
     by (rule dvd-trans)
   then show ?thesis
     by (simp add: gcd-eq-1-imp-coprime)
 qed
 — make again a pythagorean triple and obtain k and l
 from uvabc have a^2 + v^2 = u^2 by simp
 with av-relprime and aodd obtain k l where
   klavu: a = k^2−l^2 ∧ v = 2*k*l ∧ |u| = k^2+l^2 and kl-rel: coprime k l
   by (frule-tac a=a in int-euclid-pyth-triples, auto)
 — prove b = 2m and kl(k^2 + l^2) = m^2, for coprime k, l and k^2 + l^2
 from uvabc have even (b^2) by simp
 hence even b by simp
 then obtain m where bm: b = 2*m using evenE by blast
 have |k|*|l|*|k^2+l^2| = m^2
 proof −
   from bm have 4*m^2 = b^2 by (simp only: power2-eq-square ac-simps)
   also have ... = |b^2| by simp
   also with uvabc have ... = 2*|v|*||u|| by (simp add: abs-mult)
   also with klavu have ... = 2*|2*k*l|*|k^2+l^2| by simp
   also have ... = 4*|k|*|l|*|k^2+l^2| by (auto simp add: abs-mult)
   finally show ?thesis by simp
 qed
 moreover have (2::nat) > 1 by auto
 moreover from kl-rel have coprime |k| |l| by simp
 moreover have coprime |l| (|k^2+l^2|)
 proof −
   from kl-rel have coprime (k*k) l
     by simp
   hence coprime (k*k+l*l) l using gcd-add-mult [of l l k*k]
     by (simp add: ac-simps gcd-eq-1-imp-coprime)
```

```
  hence coprime l (k^2+l^2)
    by (simp add: power2-eq-square ac-simps)
  thus ?thesis by simp
qed
moreover have coprime |k^2+l^2| |k|
proof –
  from kl-rel have coprime l k
    by (simp add: ac-simps)
  hence coprime (l∗l) k
    by simp
  hence coprime (l∗l+k∗k) k using gcd-add-mult[of k k l∗l]
    by (simp add: ac-simps gcd-eq-1-imp-coprime)
  hence coprime (k^2+l^2) k
    by (simp add: power2-eq-square ac-simps)
  thus ?thesis by simp
qed
ultimately have ∃ x y z. |k| = x^2 ∧ |l| = y^2 ∧ |k^2+l^2| = z^2
  using int-relprime-power-divisors[of 2 |k| |l| ∗ |k² + l²| m]
    int-relprime-power-divisors[of 2 |l| |k| ∗ |k² + l²| m]
    int-relprime-power-divisors[of 2 |k² + l²| |k|∗|l| m]
  by (simp-all add: ac-simps)
then obtain α β γ where albega:
  |k| = α^2 ∧ |l| = β^2 ∧ |k^2+l^2| = γ^2
  by auto
— show this is a new solution
have k^2 = α^4
proof –
  from albega have |k|^2 = (α^2)^2 by simp
  thus ?thesis by simp
qed
moreover have l^2 = β^4
proof –
  from albega have |l|^2 = (β^2)^2 by simp
  thus ?thesis by simp
qed
moreover have gamma2: k^2 + l^2 = γ^2
proof –
  have k^2 ≥ 0 ∧ l^2 ≥ 0 by simp
  with albega show ?thesis by auto
qed
ultimately have newabc: α^4 + β^4 = γ^2 by auto
from uv0 klavu albega have albega0: α ∗ β ∗ γ ≠ 0 by auto
— show the coprimality
have alphabeta-relprime: coprime α β
proof (rule classical)
  let ?g = gcd α β
  assume ¬ coprime α β
  then have gnot1: ?g ≠ 1
    by (auto intro: gcd-eq-1-imp-coprime)
  have ?g > 1
  proof –
    have ?g ≠ 0
```

```
    proof
      assume ?g=0
      hence nat |α|=0 by simp
      hence α=0 by arith
      with albega0 show False by simp
    qed
    hence ?g>0 by auto
    with gnot1 show ?thesis by linarith
  qed
  moreover have ?g dvd gcd k l
  proof −
    have ?g dvd α ∧ ?g dvd β by auto
    with albega have ?g dvd |k| ∧ ?g dvd |l|
      by (simp add: power2-eq-square mult.commute)
    hence ?g dvd k ∧ ?g dvd l by simp
    thus ?thesis by simp
  qed
  ultimately have gcd k l ≠ 1 by fastforce
  with kl-rel show ?thesis by auto
qed
— choose p and q in the right way
have ∃ p q. p^4 + q^4 = γ^2 ∧ p*q*γ ≠ 0 ∧ odd p ∧ coprime p q
proof −
  have odd α ∨ odd β
  proof (rule ccontr)
    assume ¬ (odd α ∨ odd β)
    hence even α ∧ even β by simp
    then have 2 dvd α ∧ 2 dvd β by simp
    then have 2 dvd gcd α β by simp
    with alphabeta-relprime show False by auto
  qed
  moreover
  { assume odd α
    with newabc albega0 alphabeta-relprime obtain p q where
      p=α ∧ q=β ∧ p^4 + q^4 = γ^2 ∧ p*q*γ  ≠ 0 ∧ odd p ∧ coprime p q
      by auto
    hence ?thesis by auto }
  moreover
  { assume odd β
    with newabc albega0 alphabeta-relprime obtain p q where
      q=α ∧ p=β ∧ p^4 + q^4 = γ^2 ∧ p*q*γ  ≠ 0 ∧ odd p ∧ coprime p q
      by (auto simp add: ac-simps)
    hence ?thesis by auto }
  ultimately show ?thesis by auto
qed
— show the solution is smaller
moreover have γ^2 < c^2
proof −
  from gamma2 klavu have γ^2 ≤ |u| by simp
  also have h1: ... ≤ |u|^2 using self-le-power[of |u| 2] uv0 by auto
  also have h2: ... ≤ u^2 by simp
  also have h3: ... < u^2 + v^2
```

```
    proof -
      from uv0 have v2non0: 0 ≠ v^2
        by simp
      have 0 ≤ v^2 by (rule zero-le-power2)
      with v2non0 have 0 < v^2 by (auto simp add: less-le)
      thus ?thesis by auto
    qed
    also with uvabc have ... ≤ |c| by auto
    also have ... ≤ |c|^2 using self-le-power[of |c| 2] h1 h2 h3 uvabc by linarith
    also have ... ≤ c^2 by simp
    finally show ?thesis by simp
  qed
  ultimately show ?thesis by auto
qed
```

Show that no solution exists, by infinite descent of c^2.

```
private lemma no-rewritten-fermat4:
  ¬ (∃ (a::int) b. (a^4 + b^4 = c^2 ∧ a*b*c ≠ 0 ∧ odd a ∧ coprime a b))
proof (induct c rule: infinite-descent0-measure[where V=λc. nat(c^2)])
  case (0 x)
  have x^2 ≥ 0 by (rule zero-le-power2)
  with 0 have int(nat(x^2)) = 0 by auto
  hence x = 0 by auto
  thus ?case by auto
next
  case (smaller x)
  then obtain a b where a^4 + b^4 = x^2 and a*b*x ≠ 0
    and odd a and coprime a b by auto
  hence ∃ p q r. (p^4+q^4=r^2 ∧ p*q*r ≠ 0 ∧ odd p
    ∧ coprime p q ∧ r^2 < x^2) by (rule smaller-fermat4)
  then obtain p q r where pqr: p^4 + q^4 = r^2 ∧ p*q*r ≠ 0 ∧ odd p
    ∧ coprime p q ∧ r^2 < x^2 by auto
  have r^2 ≥ 0 and x^2 ≥ 0 by (auto simp only: zero-le-power2)
  hence int(nat(r^2)) = r^2 ∧ int(nat(x^2)) = x^2 by auto
  with pqr have int(nat(r^2)) < int(nat(x^2)) by auto
  hence nat(r^2) < nat(x^2) by presburger
  with pqr show ?case by auto
qed
```

The theorem. Puts equation in requested shape.

```
theorem fermat-4:
  assumes ass: (x::int)^4 + y^4 = z^4
  shows x*y*z=0
proof (rule ccontr)
  let ?g = gcd x y
  let ?c = (z div ?g)^2
  assume xyz0: x*y*z ≠ 0
  — divide out the g.c.d.
  hence x ≠ 0 ∨ y ≠ 0 by simp
  then obtain a b where ab: x = ?g*a ∧ y = ?g*b ∧ coprime a b
    using gcd-coprime-exists[of x y] by (auto simp: mult.commute)
  moreover have abc: a^4 + b^4 = ?c^2 ∧ a*b*?c ≠ 0
```

```
proof -
  have zgab: z^4 = ?g^4 * (a^4+b^4)
  proof -
    from ab ass have z^4 = (?g*a)^4+(?g*b)^4 by simp
    thus ?thesis by (simp only: power-mult-distrib distrib-left)
  qed
  have cgz: z^2 = ?c * ?g^2
  proof -
    from zgab have ?g^4 dvd z^4 by simp
    hence ?g dvd z by simp
    hence (z div ?g)*?g = z by (simp only: ac-simps dvd-mult-div-cancel)
    with ab show ?thesis by (auto simp only: power2-eq-square ac-simps)
  qed
  with xyz0 have c0: ?c≠0 by (auto simp add: power2-eq-square)
  from xyz0 have g0: ?g≠0 by simp
  have a^4 + b^4 = ?c^2
  proof -
    have ?c^2 * ?g^4 = (a^4+b^4)*?g^4
    proof -
      have ?c^2 * ?g^4 = (?c*?g^2)^2 by algebra
      also with cgz have ... = (z^2)^2 by simp
      also have ... = z^4 by algebra
      also with zgab have ... = ?g^4*(a^4+b^4) by simp
      finally show ?thesis by simp
    qed
    with g0 show ?thesis by auto
  qed
  moreover from ab xyz0 c0 have a*b*?c≠0 by auto
  ultimately show ?thesis by simp
qed
— choose the parity right
have ∃ p q. p^4 + q^4 = ?c^2 ∧ p*q*?c≠0 ∧ odd p ∧ coprime p q
proof -
  have odd a ∨ odd b
  proof (rule ccontr)
    assume ¬(odd a ∨ odd b)
    hence 2 dvd a ∧ 2 dvd b by simp
    hence 2 dvd gcd a b by simp
    with ab show False by auto
  qed
  moreover
  { assume odd a
    then obtain p q where p = a and q = b and odd p by simp
    with ab abc have ?thesis by auto }
  moreover
  { assume odd b
    then obtain p q where p = b and q = a and odd p by simp
    with ab abc have
      p^4 + q^4 = ?c^2 ∧ p*q*?c≠0 ∧ odd p ∧ coprime p q
      by (simp add: ac-simps)
    hence ?thesis by auto }
  ultimately show ?thesis by auto
```

```
  qed
  — show contradiction using the earlier result
  thus False by (auto simp only: no-rewritten-fermat4)
qed

corollary fermat-mult4:
  assumes xyz: (x::int)^n + y^n = z^n and n: 4 dvd n
  shows x*y*z=0
proof −
  from n obtain m where n = m*4 by (auto simp only: ac-simps dvd-def)
  with xyz have (x^m)^4 + (y^m)^4 = (z^m)^4 by (simp only: power-mult)
  hence (x^m)*(y^m)*(z^m) = 0 by (rule fermat-4)
  thus ?thesis by auto
qed

end

end
```

2 The quadratic form $x^2 + Ny^2$

```
theory Quad-Form
imports
  HOL−Number-Theory.Number-Theory
begin

context
begin
```

Shows some properties of the quadratic form x^2+Ny^2, such as how to multiply and divide them. The second part focuses on the case $N = 3$ and is used in the proof of the case $n = 3$ of Fermat's last theorem. The last part – not used for FLT3 – shows which primes can be written as $x^2 + 3y^2$.

2.1 Definitions and auxiliary results

```
private lemma best-division-abs: (n::int) > 0 ⟹ ∃ k. 2 * |a − k*n| ≤ n
proof −
  assume a: n > 0
  define k where k = a div n
  have h: a − k * n = a mod n by (simp add: div-mult-mod-eq algebra-simps k-def)
  thus ?thesis
  proof (cases 2 * (a mod n) ≤ n)
    case True
    hence 2 * |a − k*n| ≤ n using h pos-mod-sign a by auto
    thus ?thesis by blast
  next
    case False
    hence 2 * (n − a mod n) ≤ n by auto
    have a − (k+1)*n = a mod n − n using h by (simp add: algebra-simps)
    hence 2 * |a − (k+1)*n| ≤ n using h pos-mod-bound[of n a] a False by fastforce
```

```
    thus ?thesis by blast
  qed
qed

lemma prime-power-dvd-cancel-right:
  p ^ n dvd a if prime (p::'a::semiring-gcd) ¬ p dvd b p ^ n dvd a * b
proof -
  from that have coprime p b
    by (auto intro: prime-imp-coprime)
  with that show ?thesis
    by (simp add: coprime-dvd-mult-left-iff)
qed

definition
  is-qfN :: int ⇒ int ⇒ bool where
  is-qfN A N ⟷ (∃ x y. A = x^2 + N*y^2)

definition
  is-cube-form :: int ⇒ int ⇒ bool where
  is-cube-form a b ⟷ (∃ p q. a = p^3 − 9*p*q^2 ∧ b = 3*p^2*q − 3*q^3)

private lemma abs-eq-impl-unitfactor: |a::int| = |b| ⟹ ∃ u. a = u*b ∧ |u|=1
proof -
  assume |a| = |b|
  hence a = 1*b ∨ a = (−1)*b by arith
  then obtain u where a = u*b ∧ (u=1 ∨ u=−1) by blast
  thus ?thesis by auto
qed

private lemma prime-3-nat: prime (3::nat) by auto
```

2.2 Basic facts if $N \geq 1$

```
lemma qfN-pos: ⟦ N ≥ 1; is-qfN A N ⟧ ⟹ A ≥ 0
proof -
  assume N: N ≥ 1 and is-qfN A N
  then obtain a b where ab: A = a^2 + N*b^2 by (auto simp add: is-qfN-def)
  have N*b^2 ≥ 0
  proof (cases)
    assume b = 0 thus ?thesis by auto
  next
    assume ¬ b = 0 hence b^2 > 0 by simp
    moreover from N have N>0 by simp
    ultimately have N*b^2 > N*0 by (auto simp only: zmult-zless-mono2)
    thus ?thesis by auto
  qed
  with ab have A ≥ a^2 by auto
  moreover have a^2 ≥ 0 by (rule zero-le-power2)
  ultimately show ?thesis by arith
qed

lemma qfN-zero: ⟦ (N::int) ≥ 1; a^2 + N*b^2 = 0 ⟧ ⟹ (a = 0 ∧ b = 0)
```

```
proof −
  assume N: N ≥ 1 and abN: a^2 + N*b^2 = 0
  show ?thesis
  proof (rule ccontr, auto)
    assume a ≠ 0 hence a^2 > 0 by simp
    moreover have N*b^2 ≥ 0
    proof (cases)
      assume b = 0 thus ?thesis by auto
    next
      assume ¬ b = 0 hence  b^2 > 0 by simp
      moreover from N have N>0 by simp
      ultimately have N*b^2 > N*0 by (auto simp only: zmult-zless-mono2)
      thus ?thesis by auto
    qed
    ultimately have a^2 + N*b^2 > 0 by arith
    with abN show False by auto
  next
    assume b ≠ 0 hence b^2>0 by simp
    moreover from N have N>0 by simp
    ultimately have N*b^2>N*0 by (auto simp only: zmult-zless-mono2)
    hence N*b^2 > 0 by simp
    moreover have a^2 ≥ 0 by (rule zero-le-power2)
    ultimately have a^2 + N*b^2 > 0 by arith
    with abN show False by auto
  qed
qed
```

2.3 Multiplication and division

```
lemma qfN-mult1: ((a::int)^2 + N*b^2)*(c^2 + N*d^2)
  = (a*c+N*b*d)^2 + N*(a*d−b*c)^2
  by (simp add: eval-nat-numeral field-simps)

lemma qfN-mult2: ((a::int)^2 + N*b^2)*(c^2 + N*d^2)
  = (a*c−N*b*d)^2 + N*(a*d+b*c)^2
  by (simp add: eval-nat-numeral field-simps)

corollary is-qfN-mult: is-qfN A N ⟹ is-qfN B N ⟹ is-qfN (A*B) N
  by (unfold is-qfN-def, auto, auto simp only: qfN-mult1)

corollary is-qfN-power: (n::nat) > 0 ⟹ is-qfN A N ⟹ is-qfN (A^n) N
  by (induct n, auto, case-tac n=0, auto simp add: is-qfN-mult)

lemma qfN-div-prime:
  fixes p :: int
  assumes ass: prime (p^2+N*q^2) ∧ (p^2+N*q^2) dvd (a^2+N*b^2)
  shows ∃ u v. a^2+N*b^2 = (u^2+N*v^2)*(p^2+N*q^2)
              ∧ (∃ e. a = p*u+e*N*q*v ∧ b = p*v − e*q*u ∧ |e|=1)
proof −
  let ?P = p^2+N*q^2
  let ?A = a^2+N*b^2
  from ass obtain U where U: ?A = ?P*U by (auto simp only: dvd-def)
```

```
have ∃ e. ?P dvd b∗p + e∗a∗q ∧ |e| = 1
proof −
  have ?P dvd (b∗p + a∗q)∗(b∗p − a∗q)
  proof −
    have (b∗p + a∗q)∗(b∗p − a∗q)= b^2∗?P − q^2∗?A
      by (simp add: eval-nat-numeral field-simps)
    also from U have . . . = (b^2 − q^2∗U)∗?P by (simp add: field-simps)
    finally show ?thesis by simp
  qed
  with ass have ?P dvd (b∗p + a∗q) ∨ ?P dvd (b∗p − a∗q)
    by (simp add: nat-abs-mult-distrib prime-int-iff prime-dvd-mult-iff)
  moreover
  { assume ?P dvd b∗p + a∗q
    hence ?P dvd b∗p + 1∗a∗q ∧ |1| = (1::int) by simp }
  moreover
  { assume ?P dvd b∗p − a∗q
    hence ?P dvd b∗p + (−1)∗a∗q ∧ |−1| = (1::int) by simp }
  ultimately show ?thesis by blast
qed
then obtain v e where v: b∗p + e∗a∗q = ?P∗v and e: |e| = 1
  by (auto simp only: dvd-def)
have ?P dvd a∗p − e∗N∗b∗q
proof (cases)
  assume e1: e = 1
  from U have U ∗ ?P^2 = ?A ∗ ?P by (simp add: power2-eq-square)
  also with e1 have . . . = (a∗p−e∗N∗b∗q)^2 + N∗(b∗p+e∗a∗q)^2
    by (simp only: qfN-mult2 add.commute mult-1-left)
  also with v have . . . = (a∗p−e∗N∗b∗q)^2 + N∗v^2∗?P^2
    by (simp only: power-mult-distrib ac-simps)
  finally have (a∗p−e∗N∗b∗q)^2 = ?P^2∗(U−N∗v^2)
    by (simp add: ac-simps left-diff-distrib)
  hence ?P^2 dvd (a∗p − e∗N∗b∗q)^2 by (rule dvdI)
  thus ?thesis by simp
next
  assume ¬ e=1 with e have e1: e=−1 by auto
  from U have U ∗ ?P^2 = ?A ∗ ?P by (simp add: power2-eq-square)
  also with e1 have . . . = (a∗p−e∗N∗b∗q)^2 + N∗( −(b∗p+e∗a∗q))^2
    by (simp add: qfN-mult1)
  also have . . . = (a∗p−e∗N∗b∗q)^2 + N∗(b∗p+e∗a∗q)^2
    by (simp only: power2-minus)
  also with v have . . . = (a∗p−e∗N∗b∗q)^2 + N∗v^2∗?P^2
    by (simp only: power-mult-distrib ac-simps)
  finally have (a∗p−e∗N∗b∗q)^2 = ?P^2∗(U−N∗v^2)
    by (simp add: ac-simps left-diff-distrib)
  hence ?P^2 dvd (a∗p−e∗N∗b∗q)^2 by (rule dvdI)
  thus ?thesis by simp
qed
then obtain u where u: a∗p − e∗N∗b∗q = ?P∗u by (auto simp only: dvd-def)
from e have e2-1: e ∗ e = 1
  using abs-mult-self-eq [of e] by simp
have a: a = p∗u + e∗N∗q∗v
proof −
```

```
    have (p*u + e*N*q*v)*?P = p*(?P*u) + (e*N*q)*(?P*v)
      by (simp only: distrib-right ac-simps)
    also with v u have ... = p*(a*p − e*N*b*q) + (e*N*q)*(b*p + e*a*q)
      by simp
    also have ... = a*(p^2 + e*e*N*q^2)
      by (simp add: power2-eq-square distrib-left ac-simps right-diff-distrib)
    also with e2-1 have ... = a*?P by simp
    finally have (a−(p*u+e*N*q*v))*?P = 0 by auto
    moreover from ass have ?P ≠ 0 by auto
    ultimately show ?thesis by simp
  qed
  moreover have b: b = p*v−e*q*u
  proof −
    have (p*v−e*q*u)*?P = p*(?P*v) − (e*q)*(?P*u)
      by (simp only: left-diff-distrib ac-simps)
    also with v u have ... = p*(b*p+e*a*q) − e*q*(a*p−e*N*b*q) by simp
    also have ... = b*(p^2 + e*e*N*q^2)
      by (simp add: power2-eq-square distrib-left ac-simps right-diff-distrib)
    also with e2-1 have ... = b * ?P by simp
    finally have (b−(p*v−e*q*u))*?P = 0 by auto
    moreover from ass have ?P ≠ 0 by auto
    ultimately show ?thesis by simp
  qed
  moreover have ?A = (u^2 + N*v^2)*?P
  proof (cases)
    assume e=1
    with a and b show ?thesis by (simp add: qfN-mult1 ac-simps)
  next
    assume ¬ e=1 with e have e=−1 by simp
    with a and b show ?thesis by (simp add: qfN-mult2 ac-simps)
  qed
  moreover from e have |e| = 1 .
  ultimately show ?thesis by blast
qed

corollary qfN-div-prime-weak:
  ⟦ prime (p^2+N*q^2::int); (p^2+N*q^2) dvd (a^2+N*b^2) ⟧
  ⟹ ∃ u v. a^2+N*b^2 = (u^2+N*v^2)*(p^2+N*q^2)
  apply (subgoal-tac ∃ u v. a^2+N*b^2 = (u^2+N*v^2)*(p^2+N*q^2)
    ∧ (∃ e. a = p*u+e*N*q*v ∧ b = p*v − e*q*u ∧ |e|=1), blast)
  apply (rule qfN-div-prime, auto)
done

corollary qfN-div-prime-general: ⟦ prime P; P dvd A; is-qfN A N; is-qfN P N ⟧
  ⟹ ∃ Q. A = Q*P ∧ is-qfN Q N
  apply (subgoal-tac ∃ u v. A = (u^2+N*v^2)*P)
  apply (unfold is-qfN-def, auto)
  apply (simp only: qfN-div-prime-weak)
done

lemma qfN-power-div-prime:
  fixes P :: int
```

```
  assumes ass: prime P ∧ odd P ∧ P dvd A ∧ P^n = p^2+N*q^2
  ∧ A^n = a^2+N*b^2 ∧ coprime a b ∧ coprime p (N*q) ∧ n>0
  shows ∃ u v. a^2+N*b^2 = (u^2 + N*v^2)*(p^2+N*q^2) ∧ coprime u v
              ∧ (∃ e. a = p*u+e*N*q*v ∧ b = p*v−e*q*u ∧ |e| = 1)
proof −
  from ass have P dvd A ∧ n>0 by simp
  hence P^n dvd A^n by simp
  then obtain U where U: A^n = U*P^n by (auto simp only: dvd-def ac-simps)
  from ass have coprime a b
    by blast
  have ∃ e. P^n dvd b*p + e*a*q ∧ |e| = 1
  proof −
    have Pn-dvd-prod: P^n dvd (b*p + a*q)*(b*p − a*q)
    proof −
      have (b*p + a*q)*(b*p − a*q) = (b*p)^2 − (a*q)^2
        by (simp add: power2-eq-square algebra-simps)
      also have ... = b^2 * p^2 + b^2*N*q^2 − b^2*N*q^2 − a^2*q^2
        by (simp add: power-mult-distrib)
      also with ass have ... = b^2*P^n − q^2*A^n
        by (simp only: ac-simps distrib-right distrib-left)
      also with U have ... = (b^2−q^2*U)*P^n by (simp only: left-diff-distrib)
      finally show ?thesis by (simp add: ac-simps)
    qed
    have P^n dvd (b*p + a*q) ∨ P^n dvd (b*p − a*q)
    proof −
      have PdvdPn: P dvd P^n
      proof −
        from ass have ∃ m. n = Suc m by (simp add: not0-implies-Suc)
        then obtain m where n = Suc m by auto
        hence P^n = P*(P^m) by auto
        thus ?thesis by auto
      qed
      have ¬ P dvd b*p+a*q ∨ ¬ P dvd b*p−a*q
      proof (rule ccontr, simp)
        assume P dvd b*p+a*q ∧ P dvd b*p−a*q
        hence P dvd (b*p+a*q)+(b*p−a*q) ∧ P dvd (b*p+a*q)−(b*p−a*q)
          by (simp only: dvd-add, simp only: dvd-diff)
        hence P dvd 2*(b*p) ∧ P dvd 2*(a*q) by (simp only: mult-2, auto)
        with ass have (P dvd 2 ∨ P dvd b*p) ∧ (P dvd 2 ∨ P dvd a*q)
          using prime-dvd-multD by blast
        hence P dvd 2 ∨ (P dvd b*p ∧ P dvd a*q) by auto
        moreover have ¬ P dvd 2
        proof (rule ccontr, simp)
          assume pdvd2: P dvd 2
          have P ≤ 2
          proof (rule ccontr)
            assume ¬ P ≤ 2 hence Pl2: P > 2 by simp
            with pdvd2 show False by (simp add: zdvd-not-zless)
          qed
          moreover from ass have P > 1 by (simp add: prime-int-iff)
          ultimately have P=2 by auto
          with ass have odd 2 by simp
```

```
          thus False by simp
        qed
        ultimately have P dvd b*p ∧ P dvd a*q by auto
        with ass have (P dvd b ∨ P dvd p) ∧ (P dvd a ∨ P dvd q)
          using prime-dvd-multD by blast
        moreover have ¬ P dvd p ∧ ¬ P dvd q
        proof (auto dest: ccontr)
          assume Pdvdp: P dvd p
          hence P dvd p^2 by (simp only: dvd-mult power2-eq-square)
          with PdvdPn have P dvd P^n−p^2 by (simp only: dvd-diff)
          with ass have P dvd N*(q*q) by (simp add: power2-eq-square)
          with ass have h1: P dvd N ∨ P dvd (q*q) using prime-dvd-multD by blast
          moreover
          {
            assume P dvd (q*q)
            hence P dvd q using prime-dvd-multD ass by blast
          }
          ultimately have P dvd N*q by fastforce
          with Pdvdp have P dvd gcd p (N*q) by simp
          with ass show False by (simp add: prime-int-iff)
        next
          assume P dvd q
          hence PdvdNq: P dvd N*q by simp
          hence P dvd N*q*q by simp
          hence P dvd N*q^2 by (simp add: power2-eq-square ac-simps)
          with PdvdPn have P dvd P^n−N*q^2 by (simp only: dvd-diff)
          with ass have P dvd p*p by (simp add: power2-eq-square)
          with ass have P dvd p by (auto dest: prime-dvd-multD)
          with PdvdNq have P dvd gcd p (N*q) by auto
          with ass show False by (auto simp add: prime-int-iff)
        qed
        ultimately have P dvd a ∧ P dvd b by auto
        hence P dvd gcd a b by simp
        with ass show False by (auto simp add: prime-int-iff)
      qed
      moreover
      { assume ¬ P dvd b*p+a*q
        with Pn-dvd-prod and ass have P^n dvd b*p−a*q
            by (rule-tac b=b*p+a*q in prime-power-dvd-cancel-right, auto simp add:
mult.commute) }
      moreover
      { assume ¬ P dvd b*p−a*q
        with Pn-dvd-prod and ass have P^n dvd b*p+a*q
          by (rule-tac a=b*p+a*q in prime-power-dvd-cancel-right, simp) }
      ultimately show ?thesis by auto
    qed
    moreover
    { assume P^n dvd b*p + a*q
      hence P^n dvd b*p + 1*a*q ∧ |1| = (1::int) by simp }
    moreover
    { assume P^n dvd b*p − a*q
      hence P^n dvd b*p + (−1)*a*q ∧ |−1| = (1::int) by simp }
```

```
  ultimately show ?thesis by blast
qed
then obtain v e where v: b*p + e*a*q = P^n*v and e: |e| = 1
  by (auto simp only: dvd-def)
have P^n dvd a*p − e*N*b*q
proof (cases)
  assume e1: e = 1
  from U have (P^n)^2*U = A^n*P^n by (simp add: power2-eq-square ac-simps)
  also with e1 ass have ... = (a*p−e*N*b*q)^2 + N*(b*p+e*a*q)^2
    by (simp only: qfN-mult2 add.commute mult-1-left)
  also with v have ... = (a*p−e*N*b*q)^2 + (P^n)^2*(N*v^2)
    by (simp only: power-mult-distrib ac-simps)
  finally have (a*p−e*N*b*q)^2 = (P^n)^2*U − (P^n)^2*N*v^2 by simp
  also have ... = (P^n)^2 * (U − N*v^2) by (simp only: right-diff-distrib)
  finally have (P^n)^2 dvd (a*p − e*N*b*q)^2 by (rule dvdI)
  thus ?thesis by simp
next
  assume ¬ e=1 with e have e1: e=−1 by auto
  from U have (P^n)^2 * U = A^n * P^n by (simp add: power2-eq-square)
  also with e1 ass have ... = (a*p−e*N*b*q)^2 + N*( −(b*p+e*a*q))^2
    by (simp add: qfN-mult1)
  also have ... = (a*p−e*N*b*q)^2 + N*(b*p+e*a*q)^2
    by (simp only: power2-minus)
  also with v and ass have ... = (a*p−e*N*b*q)^2 + N*v^2*(P^n)^2
    by (simp only: power-mult-distrib ac-simps)
  finally have (a*p−e*N*b*q)^2 = (P^n)^2*U−(P^n)^2*N*v^2 by simp
  also have ... = (P^n)^2 * (U − N*v^2) by (simp only: right-diff-distrib)
  finally have (P^n)^2 dvd (a*p−e*N*b*q)^2 by (rule dvdI)
  thus ?thesis by simp
qed
then obtain u where u: a*p − e*N*b*q = P^n*u by (auto simp only: dvd-def)
from e have e2-1: e * e = 1
  using abs-mult-self-eq [of e] by simp
have a: a = p*u + e*N*q*v
proof −
  from ass have (p*u + e*N*q*v)*P^n = p*(P^n*u) + (e*N*q)*(P^n*v)
    by (simp only: distrib-right ac-simps)
  also with v and u have ... = p*(a*p − e*N*b*q) + (e*N*q)*(b*p + e*a*q)
    by simp
  also have ... = a*(p^2 + e*e*N*q^2)
    by (simp add: power2-eq-square distrib-left ac-simps right-diff-distrib)
  also with e2-1 and ass have ... = a*P^n by simp
  finally have (a−(p*u+e*N*q*v))*P^n = 0 by auto
  moreover from ass have P^n ≠ 0
    by (unfold prime-int-iff, auto)
  ultimately show ?thesis by auto
qed
moreover have b: b = p*v−e*q*u
proof −
  from ass have (p*v−e*q*u)*P^n = p*(P^n*v) − (e*q)*(P^n*u)
    by (simp only: left-diff-distrib ac-simps)
  also with v u have ... = p*(b*p+e*a*q) − e*q*(a*p−e*N*b*q) by simp
```

```
    also have ... = b*(p^2 + e*e*N*q^2)
      by (simp add: power2-eq-square distrib-left ac-simps right-diff-distrib)
    also with e2-1 and ass have ... = b * P^n by simp
    finally have (b−(p*v−e*q*u))*P^n = 0 by auto
    moreover from ass have P^n ≠ 0
      by (unfold prime-int-iff, auto)
    ultimately show ?thesis by auto
  qed
  moreover have A^n = (u^2 + N*v^2)*P^n
  proof (cases)
    assume e=1
    with a and b and ass show ?thesis by (simp add: qfN-mult1 ac-simps)
  next
    assume ¬ e=1 with e have e=−1 by simp
    with a and b and ass show ?thesis by (simp add: qfN-mult2 ac-simps)
  qed
  moreover have coprime u v
    using ‹coprime a b›
  proof (rule coprime-imp-coprime)
    fix w
    assume w dvd u w dvd v
    then have w dvd u*p + v*(e*N*q) ∧ w dvd v*p − u*(e*q)
      by simp
    with a b show w dvd a w dvd b
      by (auto simp only: ac-simps)
  qed
  moreover from e and ass have
    |e| = 1 ∧ A^n = a^2+N*b^2 ∧ P^n = p^2+N*q^2 by simp
  ultimately show ?thesis by auto
qed

lemma qfN-primedivisor-not:
  assumes ass: prime P ∧ Q > 0 ∧ is-qfN (P*Q) N ∧ ¬ is-qfN P N
  shows ∃ R. (prime R ∧ R dvd Q ∧ ¬ is-qfN R N)
proof (rule ccontr, auto)
  assume ass2: ∀ R. R dvd Q ⟶ prime R ⟶ is-qfN R N
  define ps where ps = prime-factorization (nat Q)
  from ass have ps: (∀ p∈set-mset ps. prime p) ∧ Q = int (∏ i∈#ps. i)
    by (auto simp: ps-def prod-mset-prime-factorization-int)
  have ps-lemma: ((∀ p∈set-mset ps. prime p) ∧ is-qfN (P*int(∏ i∈#ps. i)) N
    ∧ (∀ R. (prime R ∧ R dvd int(∏ i∈#ps. i)) ⟶ is-qfN R N)) ⟹ False
    (is ?B ps ⟹ False)
  proof (induct ps)
    case empty hence is-qfN P N by simp
    with ass show False by simp
  next
    case (add p ps)
    hence ass3: ?B ps ⟹ False
      and IH: ?B (ps + {#p#}) by simp-all
    hence p: prime (int p) and int p dvd int(∏ i∈#ps + {#p#}. i) by auto
    moreover with IH have pqfN: is-qfN (int p) N
     and int p dvd P*int(∏ i∈#ps + {#p#}. i) and is-qfN (P*int(∏ i∈#ps + {#p#}.
```

i)) *N*
  by *auto*
 ultimately obtain *S* **where** *S*: *P*∗*int*(∏ *i*∈#*ps* + {#*p*#}. *i*) = *S*∗(*int p*) ∧ *is-qfN S N*
  using *qfN-div-prime-general* **by** *blast*
 hence (*int p*)∗(*P*∗ *int*(∏ *i*∈#*ps*. *i*) − *S*) = *0* **by** *auto*
 with *p S* **have** *is-qfN* (*P*∗*int*(∏ *i*∈#*ps*. *i*)) *N* **by** (*auto simp add*: *prime-int-iff*)
 moreover from *IH* **have** (∀ *p*∈*set-mset ps*. *prime p*) **by** *simp*
 moreover from *IH* **have** ∀ *R*. *prime R* ∧ *R dvd int*(∏ *i*∈#*ps*. *i*) ⟶ *is-qfN R N* **by** *auto*
 ultimately have *?B ps* **by** *simp*
 with *ass3* **show** *False* **by** *simp*
qed
with *ps ass2 ass* **show** *False* **by** *auto*
qed

lemma *prime-factor-int*:
 fixes *k* :: *int*
 assumes |*k*| ≠ *1*
 obtains *p* **where** *prime p p dvd k*
proof (*cases k* = *0*)
 case *True*
 then have *prime* (*2*::*int*) **and** *2 dvd k*
  by *simp-all*
 with *that* **show** *thesis*
  by *blast*
next
 case *False*
 with *assms prime-divisor-exists* [*of k*] **obtain** *p* **where** *prime p p dvd k*
  by *auto*
 with *that* **show** *thesis*
  by *blast*
qed

lemma *qfN-oddprime-cube*:
 ⟦ *prime* (*p*^*2*+*N*∗*q*^*2*::*int*); *odd* (*p*^*2*+*N*∗*q*^*2*); *p* ≠ *0*; *N* ≥ *1* ⟧
 ⟹ ∃ *a b*. (*p*^*2*+*N*∗*q*^*2*)^*3* = *a*^*2* + *N*∗*b*^*2* ∧ *coprime a* (*N*∗*b*)
proof −
 let *?P* = *p*^*2*+*N*∗*q*^*2*
 assume *P*: *prime ?P* **and** *Podd*: *odd ?P* **and** *p0*: *p* ≠ *0* **and** *N1*: *N* ≥ *1*
 have *suc23*: *3* = *Suc 2* **by** *simp*
 let *?a* = *p*∗(*p*^*2* − *3*∗*N*∗*q*^*2*)
 let *?b* = *q*∗(*3*∗*p*^*2* − *N*∗*q*^*2*)
 have *abP*: *?P*^*3* = *?a*^*2* + *N*∗*?b*^*2* **by** (*simp add*: *eval-nat-numeral field-simps*)
 have *?P dvd p* **if** *h1*: *gcd ?b ?a* ≠ *1*
 proof −
  let *?h* = *gcd ?b ?a*
  have *h2*: *?h* ≥ *0* **by** *simp*
  hence *?h* = *0* ∨ *?h* = *1* ∨ *?h* > *1* **by** *arith*
  with *h1* **have** *?h* =*0* ∨ *?h* >*1* **by** *auto*
  moreover
  { **assume** *?h* = *0*

```
  hence ?a = 0 ∧ ?b = 0
    by auto
  with abP have ?P^3 = 0
    by auto
  with P have False
    by (unfold prime-int-iff, auto)
  hence ?thesis by simp }
moreover
{ assume ?h > 1
  then have ∃ g. prime g ∧ g dvd ?h
    using prime-factor-int [of ?h] by auto
  then obtain g where g: prime g g dvd ?h
    by blast
  then have g dvd ?b ∧ g dvd ?a by simp
  with g have g1: g dvd q ∨ g dvd 3*p^2−N*q^2
    and g2: g dvd p ∨ g dvd p^2 − 3*N*q^2
    by (auto dest: prime-dvd-multD)
  from g have gpos: g ≥ 0 by (auto simp only: prime-int-iff)
  have g dvd ?P
  proof (cases)
    assume g dvd q
    hence gNq: g dvd N*q^2 by (auto simp add: dvd-def power2-eq-square)
    show ?thesis
    proof (cases)
      assume gp: g dvd p
      hence g dvd p^2 by (auto simp add: dvd-def power2-eq-square)
      with gNq show ?thesis by auto
    next
      assume ¬ g dvd p with g2 have g dvd p^2 − 3*N*q^2 by auto
      moreover from gNq have g dvd 4*(N*q^2) by (rule dvd-mult)
      ultimately have g dvd p^2 − 3*(N*q^2) + 4*(N*q^2)
        by (simp only: ac-simps dvd-add)
      moreover have p^2 − 3*(N*q^2)+4*(N*q^2) = p^2 + N*q^2 by arith
      ultimately show ?thesis by simp
    qed
  next
    assume ¬ g dvd q with g1 have gpq: g dvd 3*p^2−N*q^2 by simp
    show ?thesis
    proof (cases)
      assume g dvd p
      hence g dvd 4*p^2 by (auto simp add: dvd-def power2-eq-square)
      with gpq have  g dvd 4*p^2 − (3*p^2 − N*q^2) by (simp only: dvd-diff)
      moreover have 4*p^2 − (3*p^2 − N*q^2) = p^2 + N*q^2 by arith
      ultimately show ?thesis by simp
    next
      assume ¬ g dvd p with g2 have g dvd p^2 − 3*N*q^2 by auto
      with gpq have g dvd 3*p^2−N*q^2 − (p^2 − 3*N*q^2)
        by (simp only: dvd-diff)
      moreover have 3*p^2−N*q^2 − (p^2 − 3*N*q^2) = 2*?P by auto
      ultimately have g dvd 2*?P by simp
      with g have g dvd 2 ∨ g dvd ?P by (simp only: prime-dvd-multD)
      moreover have ¬ g dvd 2
```

```
    proof (rule ccontr, simp)
      assume gdvd2: g dvd 2
      have g ≤ 2
      proof (rule ccontr)
        assume ¬ g ≤ 2 hence g > 2 by simp
        moreover have (0::int) < 2 by auto
        ultimately have ¬ g dvd 2 by (auto simp only: zdvd-not-zless)
        with gdvd2 show False by simp
      qed
      moreover from g have g ≥ 2 by (simp add: prime-int-iff)
      ultimately have g = 2 by auto
      with g have 2 dvd ?a ∧ 2 dvd ?b by auto
      hence 2 dvd ?a^2 ∧ 2 dvd N*?b^2
        by (simp add: power2-eq-square)
      with abP have 2 dvd ?P^3 by (simp only: dvd-add)
      hence even (?P^3) by auto
      moreover have odd (?P^3) using Podd by simp
      ultimately show False by auto
    qed
    ultimately show ?thesis by simp
  qed
qed
with P gpos have g = 1 ∨ g = ?P
  by (simp add: prime-int-iff)
with g have g = ?P by (simp add: prime-int-iff)
with g have Pab: ?P dvd ?a ∧ ?P dvd ?b by auto
have ?thesis
proof −
  from Pab P have ?P dvd p ∨ ?P dvd p^2− 3*N*q^2
    by (auto dest: prime-dvd-multD)
  moreover
  { assume ?P dvd p^2 − 3*N*q^2
    moreover have ?P dvd 3*(p^2 + N*q^2)
      by (auto simp only: dvd-refl dvd-mult)
    ultimately have ?P dvd p^2− 3*N*q^2 + 3*(p^2+N*q^2)
      by (simp only: dvd-add)
    hence ?P dvd 4*p^2 by auto
    with P have ?P dvd 4 ∨ ?P dvd p^2
      by (simp only: prime-dvd-multD)
    moreover have ¬ ?P dvd 4
    proof (rule ccontr, simp)
      assume Pdvd4: ?P dvd 4
      have ?P ≤ 4
      proof (rule ccontr)
        assume ¬ ?P ≤ 4 hence ?P > 4 by simp
        moreover have (0::int) < 4 by auto
        ultimately have ¬ ?P dvd 4 by (auto simp only: zdvd-not-zless)
        with Pdvd4 show False by simp
      qed
      moreover from P have ?P ≥ 2 by (auto simp add: prime-int-iff)
      moreover have ?P ≠ 2 ∧ ?P ≠ 4
      proof (rule ccontr, simp)
```

```
          assume ?P = 2 ∨ ?P = 4 hence even ?P by fastforce
          with Podd show False by blast
        qed
        ultimately have ?P = 3 by auto
        with Pdvd4 have (3::int) dvd 4 by simp
        thus False by arith
      qed
      ultimately have ?P dvd p*p by (simp add: power2-eq-square)
      with P have ?thesis by (auto dest: prime-dvd-multD) }
    ultimately show ?thesis by auto
  qed }
  ultimately show ?thesis by blast
qed
moreover have ?P dvd p if h1: gcd N ?a ≠ 1
proof −
  let ?h = gcd N ?a
  have h2: ?h ≥ 0 by simp
  hence ?h = 0 ∨ ?h = 1 ∨ ?h > 1 by arith
  with h1 have ?h =0 ∨ ?h >1 by auto
  moreover
  { assume ?h = 0 hence N = 0 ∧ ?a = 0
      by auto
    hence N = 0 by arith
    with N1 have False by auto
    hence ?thesis by simp }
  moreover
  { assume ?h > 1
    then have ∃ g. prime g ∧ g dvd ?h
      using prime-factor-int [of ?h] by auto
    then obtain g where g: prime g g dvd ?h
      by blast
    hence gN: g dvd N and g dvd ?a by auto
    hence g dvd p*p^2 − N*(3*p*q^2)
      by (auto simp only: right-diff-distrib ac-simps)
    with gN have g dvd p*p^2 − N*(3*p*q^2) + N*(3*p*q^2)
      by (simp only: dvd-add dvd-mult2)
    hence g dvd p*p^2 by simp
    with g have g dvd p ∨ g dvd p*p
      by (simp add: prime-dvd-multD power2-eq-square)
    with g have gp: g dvd p by (auto dest: prime-dvd-multD)
    hence g dvd p^2 by (simp add: power2-eq-square)
    with gN have gP: g dvd ?P by auto
    from g have g ≥ 0 by (simp add: prime-int-iff)
    with gP P g have g = 1 ∨ g = ?P
      by (auto dest: primes-dvd-imp-eq)
    with g have g = ?P by (auto simp only: prime-int-iff)
    with gp have ?thesis by simp }
  ultimately show ?thesis by auto
qed
moreover have ¬ ?P dvd p
proof (rule ccontr, clarsimp)
  assume Pdvdp: ?P dvd p
```

```
    have p^2 ≥ ?P^2
    proof (rule ccontr)
      assume ¬ p^2 ≥ ?P^2 hence pP: p^2 < ?P^2 by simp
      moreover with p0 have p^2 > 0 by simp
      ultimately have ¬ ?P^2 dvd p^2 by (simp add: zdvd-not-zless)
      with Pdvdp show False by simp
    qed
    moreover with P have ?P*1 < ?P*?P
      unfolding prime-int-iff by (auto simp only: zmult-zless-mono2)
    ultimately have p^2 > ?P by (auto simp add: power2-eq-square)
    hence neg: N*q^2 < 0 by auto
    show False
    proof −
      have is-qfN (0^2 + N*q^2) N by (auto simp only: is-qfN-def)
      with N1 have 0^2 +N*q^2 ≥ 0 by (rule qfN-pos)
      with neg show False by simp
    qed
  qed
  ultimately have gcd ?a ?b = 1 gcd ?a N = 1
    by (auto simp add: ac-simps)
  then have coprime ?a ?b coprime ?a N
    by (auto simp only: gcd-eq-1-imp-coprime)
  then have coprime ?a (N * ?b)
    by simp
  with abP show ?thesis
    by blast
qed
```

2.4 Uniqueness ($N > 1$)

```
lemma qfN-prime-unique:
  ⟦ prime (a^2+N*b^2::int); N > 1; a^2+N*b^2 = c^2+N*d^2 ⟧
  ⟹ (|a| = |c| ∧ |b| = |d|)
proof −
  let ?P = a^2+N*b^2
  assume P: prime ?P and N: N > 1 and abcdN: ?P = c^2 + N*d^2
  have mult: (a*d+b*c)*(a*d−b*c) = ?P*(d^2−b^2)
  proof −
    have (a*d+b*c)*(a*d−b*c) = (a^2 + N*b^2)*d^2 − b^2*(c^2 + N*d^2)
      by (simp add: eval-nat-numeral field-simps)
    with abcdN show ?thesis by (simp add: field-simps)
  qed
  have ?P dvd a*d+b*c ∨ ?P dvd a*d−b*c
  proof −
    from mult have ?P dvd (a*d+b*c)*(a*d−b*c) by simp
    with P show ?thesis by (auto dest: prime-dvd-multD)
  qed
  moreover
  { assume ?P dvd a*d+b*c
    then obtain Q where Q: a*d+b*c = ?P*Q by (auto simp add: dvd-def)
    from abcdN have ?P^2 = (a^2 + N*b^2) * (c^2 + N*d^2)
      by (simp add: power2-eq-square)
```

```
    also have ... = (a∗c−N∗b∗d)^2 + N∗(a∗d+b∗c)^2 by (rule qfN-mult2)
    also with Q have ... = (a∗c−N∗b∗d)^2 + N∗Q^2∗?P^2
      by (simp add: ac-simps power-mult-distrib)
    also have ... ≥ N∗Q^2∗?P^2 by simp
    finally have pos: ?P^2 ≥ ?P^2∗(Q^2∗N) by (simp add: ac-simps)
    have b^2 = d^2
    proof (rule ccontr)
      assume b^2 ≠ d^2
      with P mult Q have Q ≠ 0 by (unfold prime-int-iff, auto)
      hence Q^2 > 0 by simp
      moreover with N have Q^2∗N > Q^2∗1 by (simp only: zmult-zless-mono2)
      ultimately have Q^2∗N > 1 by arith
      moreover with P have ?P^2 > 0 by (simp add: prime-int-iff)
      ultimately have ?P^2∗1 < ?P^2∗(Q^2∗N) by (simp only: zmult-zless-mono2)
      with pos show False by simp
    qed }
  moreover
  { assume ?P dvd a∗d−b∗c
    then obtain Q where Q: a∗d−b∗c = ?P∗Q by (auto simp add: dvd-def)
    from abcdN have ?P^2 = (a^2 + N∗b^2) ∗ (c^2 + N∗d^2)
      by (simp add: power2-eq-square)
    also have ... = (a∗c+N∗b∗d)^2 + N∗(a∗d−b∗c)^2 by (rule qfN-mult1)
    also with Q have ... = (a∗c+N∗b∗d)^2 + N∗Q^2∗?P^2
      by (simp add: ac-simps power-mult-distrib)
    also have ... ≥ N∗Q^2∗?P^2 by simp
    finally have pos: ?P^2 ≥ ?P^2∗(Q^2∗N) by (simp add: ac-simps)
    have b^2 = d^2
    proof (rule ccontr)
      assume b^2 ≠ d^2
      with P mult Q have Q ≠ 0 by (unfold prime-int-iff, auto)
      hence Q^2 > 0 by simp
      moreover with N have Q^2∗N > Q^2∗1 by (simp only: zmult-zless-mono2)
      ultimately have Q^2∗N > 1 by arith
      moreover with P have ?P^2 > 0 by (simp add: prime-int-iff)
      ultimately have ?P^2∗1 < ?P^2 ∗ (Q^2∗N) by (simp only: zmult-zless-mono2)
      with pos show False by simp
    qed }
  ultimately have bd: b^2 = d^2 by blast
  moreover with abcdN have a^2 = c^2 by auto
  ultimately show ?thesis by (auto simp only: power2-eq-iff)
qed

lemma qfN-square-prime:
  assumes ass:
  prime (p^2+N∗q^2::int) ∧ N>1 ∧ (p^2+N∗q^2)^2 = r^2+N∗s^2 ∧ coprime r s
  shows |r| = |p^2−N∗q^2| ∧ |s| = |2∗p∗q|
proof −
  let ?P = p^2 + N∗q^2
  let ?A = r^2 + N∗s^2
  from ass have P1: ?P > 1 by (simp add: prime-int-iff)
  from ass have APP: ?A = ?P∗?P by (simp only: power2-eq-square)
  with ass have prime ?P ∧ ?P dvd ?A by (simp add: dvdI)
```

```
  then obtain u v e where uve:
    ?A = (u^2+N*v^2)*?P ∧ r = p*u+e*N*q*v ∧ s = p*v − e*q*u ∧ |e|=1
    by (frule-tac p=p in qfN-div-prime, auto)
  with APP P1 ass have prime (u^2+N*v^2) ∧ N>1 ∧ u^2 + N*v^2 = ?P
    by auto
  hence |u| = |p| ∧ |v| = |q| by (auto dest: qfN-prime-unique)
  then obtain f g where f: u = f*p ∧ |f| = 1 and g: v = g*q ∧ |g| = 1
    by (blast dest: abs-eq-impl-unitfactor)
  with uve have r = f*p*p + (e*g)*N*q*q ∧ s = g*p*q − (e*f)*p*q by simp
  hence rs: r = f*p^2 + (e*g)*N*q^2 ∧ s = (g − e*f)*p*q
    by (auto simp only: power2-eq-square left-diff-distrib)
  moreover have s ≠ 0
  proof (rule ccontr, simp)
    assume s0: s=0
    hence gcd r s = |r| by simp
    with ass have |r| = 1 by simp
    hence r^2 = 1 by (auto simp add: power2-eq-1-iff)
    with s0 have ?A = 1 by simp
    moreover have ?P^2 > 1
    proof −
      from P1 have 1 < ?P ∧ (0::int) ≤ 1 ∧ (0::nat) < 2 by auto
      hence ?P^2 > 1^2 by (simp only: power-strict-mono)
      thus ?thesis by auto
    qed
    moreover from ass have ?A = ?P^2 by simp
    ultimately show False by auto
  qed
  ultimately have g ≠ e*f by auto
  moreover from f g uve have |g| = |e*f| unfolding abs-mult by presburger
  ultimately have gef: g = −(e*f) by arith
  from uve have e * − (e * f) = − f
    using abs-mult-self-eq [of e] by simp
 hence r = f*(p^2 − N*q^2) ∧ s = (−e*f)*2*p*q using rs gef unfolding right-diff-distrib
by auto
  hence |r| = |f| * |p^2−N*q^2|
    ∧ |s| = |e|*|f|*|2*p*q|
    by (auto simp add: abs-mult)
  with uve f g show ?thesis by (auto simp only: mult-1-left)
qed

lemma qfN-cube-prime:
  assumes ass: prime (p^2 + N*q^2::int) ∧ N > 1
  ∧ (p^2 + N*q^2)^3 = a^2 + N*b^2 ∧ coprime a b
  shows |a| = |p^3− 3*N*p*q^2| ∧ |b| = |3*p^2*q−N*q^3|
proof −
  let ?P = p^2 + N*q^2
  let ?A = a^2 + N*b^2
  from ass have coprime a b by blast
  from ass have P1: ?P > 1 by (simp add: prime-int-iff)
  with ass have APP: ?A = ?P*?P^2 by (simp add: power2-eq-square power3-eq-cube)
  with ass have prime ?P ∧ ?P dvd ?A by (simp add: dvdI)
  then obtain u v e where uve:
```

?A = (u^2+N∗v^2)∗?P ∧ a = p∗u+e∗N∗q∗v ∧ b = p∗v−e∗q∗u ∧ |e|=1
by (frule-tac p=p **in** qfN-div-prime, auto)
have coprime u v
proof (rule coprimeI)
fix c
assume c dvd u c dvd v
with uve **have** c dvd a c dvd b
by simp-all
with ‹coprime a b› **show** is-unit c
by (rule coprime-common-divisor)
qed
with P1 uve APP ass **have** prime ?P ∧ N > 1 ∧ ?P^2 = u^2+N∗v^2
∧ coprime u v **by** (auto simp add: ac-simps)
hence |u| = |p^2−N∗q^2| ∧ |v| = |2∗p∗q| **by** (rule qfN-square-prime)
then obtain f g **where** f: u = f∗(p^2−N∗q^2) ∧ |f| = 1
and g: v = g∗(2∗p∗q) ∧ |g| = 1 **by** (blast dest: abs-eq-impl-unitfactor)
with uve **have** a = p∗f∗(p^2−N∗q^2) + e∗N∗q∗g∗2∗p∗q
∧ b = p∗g∗2∗p∗q −e∗q∗f∗(p^2−N∗q^2) **by** auto
hence ab: a = f∗p∗p^2 + −f∗N∗p∗q^2 + 2∗e∗g∗N∗p∗q^2
∧ b = 2∗g∗p^2∗q − e∗f∗p^2∗q + e∗f∗N∗q∗q^2
by (auto simp add: ac-simps right-diff-distrib power2-eq-square)
from f **have** f2: f^2 = 1
using abs-mult-self-eq [of f] **by** (simp add: power2-eq-square)
from g **have** g2: g^2 = 1
using abs-mult-self-eq [of g] **by** (simp add: power2-eq-square)
have e ≠ f∗g
proof (rule ccontr, simp)
assume efg: e = f∗g
with ab g2 **have** a = f∗p∗p^2+f∗N∗p∗q^2 **by** (auto simp add: power2-eq-square)
hence a = (f∗p)∗?P **by** (auto simp add: distrib-left ac-simps)
hence Pa: ?P dvd a **by** auto
have e ∗ f = g **using** f2 power2-eq-square[of f] efg **by** simp
with ab **have** b = g∗p^2∗q+g∗N∗q∗q^2 **by** auto
hence b = (g∗q)∗?P **by** (auto simp add: distrib-left ac-simps)
hence ?P dvd b **by** auto
with Pa **have** ?P dvd gcd a b **by** simp
with ass **have** ?P dvd 1 **by** auto
with P1 **show** False **by** auto
qed
moreover from f g uve **have** |e| = |f∗g| **unfolding** abs-mult **by** auto
ultimately have e = −(f∗g) **by** arith
hence e ∗ g = −f e ∗ f = −g **using** f2 g2 **unfolding** power2-eq-square **by** auto
with ab **have** a = f∗p∗p^2 − 3∗f∗N∗p∗q^2 ∧ b = 3∗g∗p^2∗q − g∗N∗q∗q^2 **by** (simp add: mult.assoc)
hence a = f∗(p^3 − 3∗N∗p∗q^2) ∧ b = g∗(3∗p^2∗q − N∗q^3)
by (auto simp only: right-diff-distrib ac-simps power2-eq-square power3-eq-cube)
with f g **show** ?thesis **by** (auto simp add: abs-mult)
qed

2.5 The case $N = 3$

lemma qf3-even: even (a^2+3∗b^2) ⟹ ∃ B. a^2+3∗b^2 = 4∗B ∧ is-qfN B 3

```
proof −
  let ?A = a^2+3*b^2
  assume even: even ?A
  have (odd a ∧ odd b) ∨ (even a ∧ even b)
  proof (rule ccontr, auto)
    assume even a and odd b
    hence even (a^2) ∧ odd (b^2)
      by (auto simp add: power2-eq-square)
    moreover have odd 3 by simp
    ultimately have odd ?A by simp
    with even show False by simp
  next
    assume odd a and even b
    hence odd (a^2) ∧ even (b^2)
      by (auto simp add: power2-eq-square)
    moreover hence even (b^2*3) by simp
    ultimately have odd (b^2*3+a^2) by simp
    hence odd ?A by (simp add: ac-simps)
    with even show False by simp
  qed
  moreover
  { assume even a ∧ even b
    then obtain c d where abcd: a = 2*c ∧ b = 2*d using evenE[of a] evenE[of b] by
meson
    hence ?A = 4*(c^2 + 3*d^2) by (simp add: power-mult-distrib)
    moreover have is-qfN (c^2+3*d^2) 3 by (unfold is-qfN-def, auto)
    ultimately have ?thesis by blast }
  moreover
  { assume odd a ∧ odd b
    then obtain c d where abcd: a = 2*c+1 ∧ b = 2*d+1 using oddE[of a] oddE[of
b] by meson
    have odd (c−d) ∨ even (c−d) by blast
    moreover
    { assume even (c−d)
      then obtain e where c−d = 2*e using evenE by blast
      with abcd have e1: a−b = 4*e by arith
      hence e2: a+3*b = 4*(e+b) by auto
      have 4*?A = (a+3*b)^2 + 3*(a−b)^2
        by (simp add: eval-nat-numeral field-simps)
      also with e1 e2 have ... = (4*(e+b))^2+3*(4*e)^2 by (simp(no-asm-simp))
    finally have ?A = 4*((e+b)^2 + 3*e^2) by (simp add: eval-nat-numeral field-simps)
      moreover have is-qfN ((e+b)^2 + 3*e^2) 3 by (unfold is-qfN-def, auto)
      ultimately have ?thesis by blast }
    moreover
    { assume odd (c−d)
      then obtain e where c−d = 2*e+1 using oddE by blast
      with abcd have e1: a+b = 4*(e+d+1) by auto
      hence e2: a− 3*b = 4*(e+d−b+1) by auto
      have 4*?A = (a− 3*b)^2 + 3*(a+b)^2
        by (simp add: eval-nat-numeral field-simps)
      also with e1 e2 have ... = (4*(e+d−b+1))^2 +3*(4*(e+d+1))^2
        by (simp (no-asm-simp))
```

```
    finally have ?A = 4*((e+d−b+1)^2+3*(e+d+1)^2)
      by (simp add: eval-nat-numeral field-simps)
    moreover have is-qfN ((e+d−b+1)^2 + 3*(e+d+1)^2) 3
      by (unfold is-qfN-def, auto)
    ultimately have ?thesis by blast }
  ultimately have ?thesis by auto }
 ultimately show ?thesis by auto
qed

lemma qf3-even-general: ⟦ is-qfN A 3; even A ⟧
 ⟹ ∃ B. A = 4*B ∧ is-qfN B 3
proof −
 assume even A and is-qfN A 3
 then obtain a b where A = a^2 + 3*b^2
  and even (a^2 + 3*b^2) by (unfold is-qfN-def, auto)
 thus ?thesis by (auto simp add: qf3-even)
qed

lemma qf3-oddprimedivisor-not:
 assumes ass: prime P ∧ odd P ∧ Q>0 ∧ is-qfN (P*Q) 3 ∧ ¬ is-qfN P 3
 shows ∃ R. prime R ∧ odd R ∧ R dvd Q ∧ ¬ is-qfN R 3
proof (rule ccontr, simp)
 assume ass2: ∀ R. R dvd Q ⟶ prime R ⟶ even R ∨ is-qfN R 3
 (is ?A Q)
 obtain n::nat where n = nat Q by auto
 with ass have n: Q = int n by auto
 have (n > 0 ∧ is-qfN (P*int n) 3 ∧ ?A(int n)) ⟹ False (is ?B n ⟹ False)
 proof (induct n rule: less-induct)
  case (less n)
  hence IH: !!m. m<n ∧ ?B m ⟹ False
   and Bn: ?B n by auto
  show False
  proof (cases)
   assume odd: odd (int n)
   from Bn ass have prime P ∧ int n > 0 ∧ is-qfN (P*int n) 3 ∧ ¬ is-qfN P 3
    by simp
   hence ∃ R. prime R ∧ R dvd int n ∧ ¬ is-qfN R 3
    by (rule qfN-primedivisor-not)
   then obtain R where R: prime R ∧ R dvd int n ∧ ¬ is-qfN R 3 by auto
   moreover with odd have odd R
   proof −
    from R obtain U where int n = R*U by (auto simp add: dvd-def)
    with odd show ?thesis by auto
   qed
   moreover from Bn have ?A (int n) by simp
   ultimately show False by auto
  next
   assume even: ¬ odd (int n)
   hence even ((int n)*P) by simp
   with Bn have  even (P*int n) ∧ is-qfN (P*int n) 3 by (simp add: ac-simps)
   hence ∃ B. P*(int n) = 4*B ∧ is-qfN B 3 by (simp only: qf3-even-general)
   then obtain B where B: P*(int n) = 4*B ∧ is-qfN B 3 by auto
```

```
      hence 2^2 dvd (int n)*P by (simp add: ac-simps)
      moreover have ¬ 2 dvd P
      proof (rule ccontr, simp)
        assume 2 dvd P
        with ass have odd P ∧ even P by simp
        thus False by simp
      qed
      moreover have prime (2::int) by simp
      ultimately have 2^2 dvd int n
        by (rule-tac p=2 in prime-power-dvd-cancel-right)
      then obtain im::int where int n = 4*im by (auto simp add: dvd-def)
      moreover obtain m::nat where m = nat im by auto
      ultimately have m: n = 4*m by arith
      with B have is-qfN (P*int m) 3 by auto
      moreover from m Bn have m > 0 by auto
      moreover from m Bn have ?A (int m) by auto
      ultimately have Bm: ?B m by simp
      from Bn m have m < n by arith
      with IH Bm show False by auto
    qed
  qed
  with ass ass2 n show False by auto
qed

lemma qf3-oddprimedivisor:
  ⟦ prime (P::int); odd P; coprime a b; P dvd (a^2+3*b^2) ⟧
  ⟹ is-qfN P 3
proof(induct P arbitrary:a b rule:infinite-descent0-measure[where V=λP. nat|P|])
  case (0 x)
  moreover hence x = 0 by arith
  ultimately show ?case by (simp add: prime-int-iff)
next
  case (smaller x)
  then obtain a b where abx: prime x ∧ odd x ∧ coprime a b
    ∧ x dvd (a^2+3*b^2) ∧ ¬ is-qfN x 3 by auto
  then obtain M where M: a^2+3*b^2 = x*M by (auto simp add: dvd-def)
  let ?A = a^2 + 3*b^2
  from abx have x0: x > 0 by (simp add: prime-int-iff)
  then obtain m where 2*|a−m*x|≤x by (auto dest: best-division-abs)
  with abx have 2*|a−m*x|<x using odd-two-times-div-two-succ[of x] by presburger
  then obtain c where cm: c = a−m*x ∧ 2*|c| < x by auto
  from x0 obtain n where 2*|b−n*x|≤x by (auto dest: best-division-abs)
  with abx have 2*|b−n*x|<x using odd-two-times-div-two-succ[of x] by presburger
  then obtain d where dn: d = b−n*x ∧ 2*|d| < x by auto
  let ?C = c^2+3*d^2
  have C3: is-qfN ?C 3 by (unfold is-qfN-def, auto)
  have C0: ?C > 0
  proof −
    have hlp: (3::int) ≥ 1 by simp
    have ?C ≥ 0 by simp
    hence ?C = 0 ∨ ?C > 0 by arith
    moreover
```

```
  { assume ?C = 0
    with hlp have c=0 ∧ d=0 by (rule qfN-zero)
    with cm dn have a = m*x ∧ b = n*x by simp
    hence x dvd a ∧ x dvd b by simp
    hence x dvd gcd a b by simp
    with abx have False by (auto simp add: prime-int-iff) }
  ultimately show ?thesis by blast
qed
have x dvd ?C
proof
  have ?C = |c|^2 + 3*|d|^2 by (simp only: power2-abs)
  also with cm dn have ... = (a−m*x)^2 + 3*(b−n*x)^2 by simp
  also have ... =
    a^2 − 2*a*(m*x) + (m*x)^2 + 3*(b^2 − 2*b*(n*x) + (n*x)^2)
    by (simp add: algebra-simps power2-eq-square)
  also with abx M have ... =
    x*M − x*(2*a*m + 3*2*b*n) + x^2*(m^2 + 3*n^2)
    by (simp only: power-mult-distrib distrib-left ac-simps, auto)
  finally show ?C = x*(M − (2*a*m + 3*2*b*n) + x*(m^2 + 3*n^2))
    by (simp add: power2-eq-square distrib-left right-diff-distrib)
qed
then obtain y where y: ?C = x*y by (auto simp add: dvd-def)
have yx: y < x
proof (rule ccontr)
  assume ¬ y < x hence xy: x−y ≤ 0 by simp
  have hlp: 2*|c| ≥ 0 ∧ 2*|d| ≥ 0 ∧ (3::nat) > 0 by simp
  from y have 4*x*y = 2^2*c^2 + 3*2^2*d^2 by simp
  hence 4*x*y = (2*|c|)^2 + 3*(2*|d|)^2
    by (auto simp add: power-mult-distrib)
  with cm dn hlp have 4*x*y < x^2 + 3*(2*|d|)^2
    and (3::int) > 0 ∧ (2*|d|)^2 < x^2
          using power-strict-mono [of 2*|b| x 2 for b]
    by auto
  hence x*4*y < x^2 + 3*x^2 by (auto)
  also have ... = x*4*x by (simp add: power2-eq-square)
  finally have contr: (x−y)*(4*x) > 0 by (auto simp add: right-diff-distrib)
  show False
  proof (cases)
    assume x−y = 0 with contr show False by auto
  next
    assume ¬ x−y =0 with xy have x−y < 0 by simp
    moreover from x0 have 4*x > 0 by simp
    ultimately have 4*x*(x−y) < 4*x*0 by (simp only: zmult-zless-mono2)
    with contr show False by auto
  qed
qed
have y0: y > 0
proof (rule ccontr)
  assume ¬ y > 0
  hence y ≤ 0 by simp
  moreover have y ≠ 0
  proof (rule ccontr)
```

```
    assume ¬ y≠0 hence y=0 by simp
    with y and C0 show False by auto
  qed
  ultimately have y < 0 by simp
  with x0 have x*y < x*0 by (simp only: zmult-zless-mono2)
  with C0 y show False by simp
qed
let ?g = gcd c d
have c ≠ 0 ∨ d ≠ 0
proof (rule ccontr)
  assume ¬ (c≠0 ∨ d≠0) hence c=0 ∧ d=0 by simp
  with C0 show False by simp
qed
then obtain e f where ef: c = ?g*e ∧ d = ?g * f ∧ coprime e f
  using gcd-coprime-exists[of c d] gcd-pos-int[of c d] by (auto simp: mult.commute)
have g2nonzero: ?g^2 ≠ 0
proof (rule ccontr, simp)
  assume c = 0 ∧ d = 0
  with C0 show False by simp
qed
let ?E = e^2 + 3*f^2
have E3: is-qfN ?E 3 by (unfold is-qfN-def, auto)
have CgE: ?C = ?g^2 * ?E
proof −
  have ?g^2 * ?E = (?g*e)^2 + 3*(?g*f)^2
    by (simp add: distrib-left power-mult-distrib)
  with ef show ?thesis by simp
qed
hence ?g^2 dvd ?C by (simp add: dvd-def)
with y have g2dvdxy: ?g^2 dvd y*x by (simp add: ac-simps)
moreover have coprime x (?g^2)
proof −
  let ?h = gcd ?g x
  have ?h dvd ?g and ?g dvd c by blast+
  hence ?h dvd c by (rule dvd-trans)
  have ?h dvd ?g and ?g dvd d by blast+
  hence ?h dvd d by (rule dvd-trans)
  have ?h dvd x by simp
  hence ?h dvd m*x by (rule dvd-mult)
  with ‹?h dvd c› have ?h dvd c+m*x by (rule dvd-add)
  with cm have ?h dvd a by simp
  from ‹?h dvd x› have ?h dvd n*x by (rule dvd-mult)
  with ‹?h dvd d› have ?h dvd d+n*x by (rule dvd-add)
  with dn have ?h dvd b by simp
  with ‹?h dvd a› have ?h dvd gcd a b by simp
  with abx have ?h dvd 1 by simp
  hence ?h = 1 by simp
  hence coprime (?g^2) x by (auto intro: gcd-eq-1-imp-coprime)
  thus ?thesis by (simp only: ac-simps)
qed
ultimately have ?g^2 dvd y
  by (auto simp add: ac-simps coprime-dvd-mult-right-iff)
```

```
then obtain w where w: y = ?g^2 * w by (auto simp add: dvd-def)
with CgE y g2nonzero have Ewx: ?E = x*w by auto
have w>0
proof (rule ccontr)
  assume ¬ w>0 hence w ≤ 0 by auto
  hence w=0 ∨ w<0 by auto
  moreover
  { assume w=0 with w y0 have False by auto }
  moreover
  { assume wneg: w<0
    have ?g^2 ≥ 0 by (rule zero-le-power2)
    with g2nonzero have ?g^2 > 0 by arith
    with wneg have ?g^2*w < ?g^2*0 by (simp only: zmult-zless-mono2)
    with w y0 have False by auto }
  ultimately show False by blast
qed
have w-le-y: w ≤ y
proof (rule ccontr)
  assume ¬ w ≤ y
  hence wy: w > y by simp
  have ?g^2 = 1 ∨ ?g^2 > 1
  proof −
    have ?g^2 ≥ 0 by (rule zero-le-power2)
    hence ?g^2 =0 ∨ ?g^2 > 0 by auto
    with g2nonzero show ?thesis by arith
  qed
  moreover
  { assume ?g^2 =1 with w wy have False by simp }
  moreover
  { assume g1: ?g^2 >1
    with ‹w>0› have w*1 < w*?g^2 by (auto dest: zmult-zless-mono2)
    with w have w < y by (simp add: ac-simps)
    with wy have False by auto }
  ultimately show False by blast
qed
from Ewx E3 abx ‹w>0› have
  prime x ∧ odd x ∧ w > 0 ∧ is-qfN (x*w) 3 ∧ ¬ is-qfN x 3 by simp
then obtain z where z: prime z ∧ odd z ∧ z dvd w ∧ ¬ is-qfN z 3
  by (frule-tac P=x in qf3-oddprimedivisor-not, auto)
from Ewx have w dvd ?E by simp
with z have z dvd ?E by (auto dest: dvd-trans)
with z ef have prime z ∧ odd z ∧ coprime e f ∧ z dvd ?E ∧ ¬ is-qfN z 3
  by auto
moreover have nat|z| < nat|x|
proof −
  have z ≤ w
  proof (rule ccontr)
    assume ¬ z ≤ w hence w < z by auto
    with ‹w>0› have ¬ z dvd w by (rule zdvd-not-zless)
    with z show False by simp
  qed
  with w-le-y yx have z < x by simp
```

```
    with z have |z| < |x| by (simp add: prime-int-iff)
    thus ?thesis by auto
  qed
  ultimately show ?case by auto
qed

lemma qf3-cube-prime-impl-cube-form:
  assumes ab-relprime: coprime a b and abP: P^3 = a^2 + 3*b^2
  and P: prime P ∧ odd P
  shows is-cube-form a b
proof −
  from abP have qfP3: is-qfN (P^3) 3 by (auto simp only: is-qfN-def)
  have PvdP3: P dvd P^3 by (simp add: eval-nat-numeral)
  with abP ab-relprime P have qfP: is-qfN P 3 by (simp add: qf3-oddprimedivisor)
  then obtain p q where pq: P = p^2 + 3*q^2 by (auto simp only: is-qfN-def)
  with P abP ab-relprime have prime (p^2 + 3*q^2) ∧ (3::int) > 1
    ∧ (p^2+3*q^2)^3 = a^2+3*b^2 ∧ coprime a b by auto
  hence ab: |a| = |p^3 − 3*3*p*q^2| ∧ |b| = |3*p^2*q − 3*q^3|
    by (rule qfN-cube-prime)
  hence a: a = p^3 − 9*p*q^2 ∨ a = −(p^3) + 9*p*q^2 by arith
  from ab have b: b = 3*p^2*q − 3*q^3 ∨ b = −(3*p^2*q) + 3*q^3 by arith
  obtain r s where r: r = −p and s: s = −q  by simp
  show ?thesis
  proof (cases)
    assume a1: a = p^3− 9*p*q^2
    show ?thesis
    proof (cases)
      assume b1: b = 3*p^2*q − 3*q^3
      with a1 show ?thesis by (unfold is-cube-form-def, auto)
    next
      assume ¬ b = 3*p^2*q − 3*q^3
      with b have b = − 3*p^2*q + 3*q^3 by simp
      with s have b = 3*p^2*s − 3*s^3 by simp
      moreover from a1 s have a = p^3 − 9*p*s^2 by simp
      ultimately show ?thesis by (unfold is-cube-form-def, auto)
    qed
  next
    assume ¬ a = p^3 − 9*p*q^2
    with a have a = −(p^3) + 9*p*q^2 by simp
    with r have ar: a = r^3 − 9*r*q^2 by simp
    show ?thesis
    proof (cases)
      assume b1: b = 3*p^2*q − 3*q^3
      with r have b = 3*r^2*q − 3*q^3 by simp
      with ar show ?thesis by (unfold is-cube-form-def, auto)
    next
      assume ¬ b = 3*p^2*q − 3*q^3
      with b have b = − 3*p^2*q + 3*q^3 by simp
      with r s have b = 3*r^2*s − 3*s^3 by simp
      moreover from ar s have a = r^3 − 9*r*s^2 by simp
      ultimately show ?thesis by (unfold is-cube-form-def, auto)
    qed
```

```
  qed
qed

lemma cube-form-mult: ⟦ is-cube-form a b; is-cube-form c d; |e| = 1 ⟧
  ⟹ is-cube-form (a*c+e*3*b*d) (a*d−e*b*c)
proof −
  assume ab: is-cube-form a b and c-d: is-cube-form c d and e: |e| = 1
  from ab obtain p q where pq: a = p^3 − 9*p*q^2 ∧ b = 3*p^2*q − 3*q^3
    by (auto simp only: is-cube-form-def)
  from c-d obtain r s where rs: c = r^3 − 9*r*s^2 ∧ d = 3*r^2*s − 3*s^3
    by (auto simp only: is-cube-form-def)
  let ?t = p*r + e*3*q*s
  let ?u = p*s − e*r*q
  have e2: e^2=1
  proof −
    from e have e=1 ∨ e=−1 by linarith
    moreover
    { assume e=1 hence ?thesis by auto }
    moreover
    { assume e=−1 hence ?thesis by simp }
    ultimately show ?thesis by blast
  qed
  hence e*e^2 = e by simp
  hence e3: e*1 = e^3 by (simp only: power2-eq-square power3-eq-cube)
  have a*c+e*3*b*d = ?t^3 − 9*?t*?u^2
  proof −
    have ?t^3 − 9*?t*?u^2 = p^3*r^3 + e*9*p^2*q*r^2*s + e^2*27*p*q^2*r*s^2
    + e^3*27*q^3*s^3 − 9*p*p^2*r*s^2 + e*18*p^2*q*r^2*s − e^2*9*p*q^2*(r*r^2)
     − e*27*p^2*q*(s*s^2) + e^2*54*p*q^2*r*s^2 − e*e^2*27*(q*q^2)*r^2*s
      by (simp add: eval-nat-numeral field-simps)
    also with e2 e3 have ... =
      p^3*r^3 + e*27*p^2*q*r^2*s + 81*p*q^2*r*s^2 + e*27*q^3*s^3
      − 9*p^3*r*s^2 − 9*p*q^2*r^3 − e*27*p^2*q*s^3 − e*27*q^3*r^2*s
      by (simp add: power2-eq-square power3-eq-cube)
    also with pq rs have ... = a*c + e*3*b*d
      by (simp only: left-diff-distrib right-diff-distrib ac-simps)
    finally show ?thesis by auto
  qed
  moreover have a*d−e*b*c = 3*?t^2*?u − 3*?u^3
  proof −
    have 3*?t^2*?u − 3*?u^3 =
      3*(p*p^2)*r^2*s − e*3*p^2*q*(r*r^2) + e*18*p^2*q*r*s^2
      − e^2*18*p*q^2*r^2*s + e^2*27*p*q^2*(s*s^2) − e*e^2*27*(q*q^2)*r*s^2
      − 3*p^3*s^3 + e*9*p^2*q*r*s^2 − e^2*9*p*q^2*r^2*s + e^3*3*r^3*q^3
      by (simp add: eval-nat-numeral field-simps)
    also with e2 e3 have ... = 3*p^3*r^2*s − e*3*p^2*q*r^3 + e*18*p^2*q*r*s^2
     − 18*p*q^2*r^2*s + 27*p*q^2*s^3 − e*27*q^3*r*s^2 − 3*p^3*s^3
     + e*9*p^2*q*r*s^2 − 9*p*q^2*r^2*s + e*3*r^3*q^3
      by (simp add: power2-eq-square power3-eq-cube)
    also with pq rs have ... = a*d−e*b*c
      by (simp only: left-diff-distrib right-diff-distrib ac-simps)
    finally show ?thesis by auto
```

```
  qed
  ultimately show ?thesis by (auto simp only: is-cube-form-def)
qed

lemma qf3-cube-primelist-impl-cube-form: ⟦ (∀ p∈set-mset ps. prime p); odd (int (∏ i∈#ps.
i)) ⟧ ⟹
  (!! a b. coprime a b ⟹ a^2 + 3*b^2 = (int(∏ i∈#ps. i))^3 ⟹ is-cube-form a b)
proof (induct ps)
  case empty hence ab1: a^2 + 3*b^2 = 1 by simp
  have b0: b=0
  proof (rule ccontr)
    assume b≠0
    hence b^2>0 by simp
    hence 3*b^2 > 1 by arith
    with ab1 have a^2 < 0 by arith
    moreover have a^2 ≥ 0 by (rule zero-le-power2)
    ultimately show False by auto
  qed
  with ab1 have a1: (a=1 ∨ a=−1) by (auto simp add: power2-eq-square zmult-eq-1-iff)
  then obtain p and q where p=a and q=(0::int) by simp
  with a1 and b0 have a = p^3 − 9*p*q^2 ∧ b = 3*p^2*q − 3*q^3 by auto
  thus is-cube-form a b by (auto simp only: is-cube-form-def)
next
  case (add p ps) hence ass: coprime a b ∧ odd (int(∏ i∈#ps + {#p#}. i))
    ∧ a^2+3*b^2 = int(∏ i∈#ps + {#p#}. i)^3 ∧ (∀ a∈set-mset ps. prime a) ∧ prime
(int p)
    and IH: !! u v. coprime u v ∧ u^2+3*v^2 = int(∏ i∈#ps. i)^3
    ∧ odd (int(∏ i∈#ps. i)) ⟹ is-cube-form u v
    by auto
  then have coprime a b
    by simp
  let ?w = int (∏ i∈#ps + {#p#}. i)
  let ?X = int (∏ i∈#ps. i)
  let ?p = int p
  have ge3-1: (3::int) ≥ 1 by auto
  have pw: ?w = ?p * ?X ∧ odd ?p ∧ odd ?X
  proof (safe)
    have (∏ i∈#ps + {#p#}. i) = p * (∏ i∈#ps. i) by simp
    thus wpx: ?w = ?p * ?X by (auto simp only: of-nat-mult [symmetric])
    with ass show even ?p ⟹ False by auto
    from wpx have ?w = ?X*?p by simp
    with ass show even ?X ⟹ False by simp
  qed
  have is-qfN ?p 3
  proof −
    from ass have a^2+3*b^2 = (?p*?X)^3 by (simp add: mult.commute)
    hence ?p dvd a^2+3*b^2 by (simp add: eval-nat-numeral field-simps)
    moreover from ass have prime ?p and coprime a b by simp-all
    moreover from pw have odd ?p by simp
    ultimately show ?thesis by (simp add: qf3-oddprimedivisor)
  qed
  then obtain α β where alphabeta: ?p = α^2 + 3*β^2
```

```
    by (auto simp add: is-qfN-def)
  have α ≠ 0
  proof (rule ccontr, simp)
    assume α = 0 with alphabeta have 3 dvd ?p by auto
    with pw have w3: 3 dvd ?w by (simp only: dvd-mult2)
    then obtain v where ?w = 3∗v by (auto simp add: dvd-def)
    with ass have vab: 27∗v^3 = a^2 + 3∗b^2 by simp
    hence a^2 = 3∗(9∗v^3 − b^2) by auto
    hence 3 dvd a^2 by (unfold dvd-def, blast)
    moreover have prime (3::int) by simp
    ultimately have a3: 3 dvd a using prime-dvd-power-int[of 3::int a 2] by fastforce
    then obtain c where c: a = 3∗c by (auto simp add: dvd-def)
    with vab have 27∗v^3 = 9∗c^2 + 3∗b^2 by (simp add: power-mult-distrib)
    hence b^2 = 3∗(3∗v^3 − c^2) by auto
    hence 3 dvd b^2 by (unfold dvd-def, blast)
    moreover have prime (3::int) by simp
    ultimately have 3 dvd b using prime-dvd-power-int[of 3::int b 2] by fastforce
    with a3 have 3 dvd gcd a b by simp
    with ass show False by simp
  qed
  moreover from alphabeta pw ass have
    prime (α^2 + 3∗β^2) ∧ odd (α^2+3∗β^2) ∧ (3::int) ≥ 1 by auto
  ultimately obtain c d where cdp:
    (α^2+3∗β^2)^3 = c^2+3∗d^2 ∧ coprime c (3∗d)
    by (blast dest: qfN-oddprime-cube)
  with ass pw alphabeta have ∃ u v. a^2+3∗b^2 = (u^2 + 3∗v^2)∗(c^2+3∗d^2)
    ∧ coprime u v ∧ (∃ e. a = c∗u+e∗3∗d∗v ∧ b = c∗v−e∗d∗u ∧ |e| = 1)
    by (rule-tac A=?w and n=3 in qfN-power-div-prime, auto)
  then obtain u v e where uve: a^2+3∗b^2 = (u^2+3∗v^2)∗(c^2+3∗d^2)
    ∧ coprime u v ∧ a = c∗u+e∗3∗d∗v ∧ b = c∗v−e∗d∗u ∧ |e| = 1 by blast
  moreover have is-cube-form u v
  proof −
    have uvX: u^2+3∗v^2 = ?X^3
    proof −
      from ass have p0: ?p ≠ 0 by (simp add: prime-int-iff)
      from pw have ?p^3∗?X^3 = ?w^3 by (simp add: power-mult-distrib)
      also with ass have . . . = a^2+3∗b^2 by simp
      also with uve have . . . = (u^2+3∗v^2)∗(c^2+3∗d^2) by auto
     also with cdp alphabeta have . . . = ?p^3 ∗ (u^2+3∗v^2) by (simp only: ac-simps)
      finally have ?p^3∗(u^2+3∗v^2−?X^3) = 0 by auto
      with p0 show ?thesis by auto
    qed
    with pw IH uve show ?thesis by simp
  qed
  moreover have is-cube-form c d
  proof −
    have coprime c d
    proof (rule coprimeI)
      fix f
      assume f dvd c and f dvd d
      then have f dvd c∗u + d∗(e∗3∗v) ∧ f dvd c∗v−d∗(e∗u)
        by simp
```

```
      with uve have f dvd a and f dvd b
        by (auto simp only: ac-simps)
      with ‹coprime a b› show is-unit f
        by (rule coprime-common-divisor)
    qed
    with pw cdp ass alphabeta show ?thesis
      by (rule-tac P=?p in qf3-cube-prime-impl-cube-form, auto)
  qed
  ultimately show is-cube-form a b by (simp only: cube-form-mult)
qed

lemma qf3-cube-impl-cube-form:
  assumes ass: coprime a b ∧ a^2 + 3*b^2 = w^3 ∧ odd w
  shows is-cube-form a b
proof −
  have 0 ≤ w^3 using ass not-sum-power2-lt-zero[of a b] zero-le-power2[of b] by linarith
  hence 0 < w using ass by auto arith
  define M where M = prime-factorization (nat w)
  from ‹w > 0› have (∀p∈set-mset M. prime p) ∧ w = int (∏i∈#M. i)
    by (auto simp: M-def prod-mset-prime-factorization-int)
  with ass show ?thesis by (auto dest: qf3-cube-primelist-impl-cube-form)
qed
```

2.6 Existence ($N = 3$)

This part contains the proof that all prime numbers $\equiv 1$ mod 6 can be written as $x^2 + 3y^2$.

First show $(\frac{a}{p})(\frac{b}{p}) = (\frac{ab}{p})$, where p is an odd prime.

```
lemma Legendre-zmult: ⟦ p > 2; prime p ⟧
  ⟹ (Legendre (a*b) p) = (Legendre a p)*(Legendre b p)
proof −
  assume p2: p > 2 and prp: prime p
  from prp have prp': prime (nat p)
    by simp
  let ?p12 = nat(((p) − 1) div 2)
  let ?Labp = Legendre (a*b) p
  let ?Lap = Legendre a p
  let ?Lbp = Legendre b p
  have h1: ((nat p − 1) div 2) = nat ((p − 1) div 2) using p2 by auto
  hence [?Labp = (a*b)^?p12] (mod p) using prp p2 euler-criterion[of nat p a*b]
    by auto
  hence [a^?p12 * b^?p12 = ?Labp] (mod p)
    by (simp only: power-mult-distrib cong-sym)
  moreover have [?Lap * ?Lbp = a^?p12*b^?p12] (mod p)
    using euler-criterion[of nat p] p2 prp' h1 by (simp add: cong-mult)
  ultimately have [?Lap * ?Lbp = ?Labp] (mod p)
    using cong-trans by blast
  then obtain k where k: ?Labp = (?Lap*?Lbp) + p * k
    by (auto simp add: cong-iff-lin)
  have k=0
  proof (rule ccontr)
```

```
    assume k ≠ 0 hence |k| = 1 ∨ |k| > 1 by arith
    moreover
    { assume |k|= 1
      with p2 have |k|*p > 2 by auto }
    moreover
    { assume k1: |k| > 1
      with p2 have |k|*2 < |k|*p
        by (simp only: zmult-zless-mono2)
      with k1 have |k|*p > 2 by arith }
   ultimately have |k|*p > 2 by auto
   moreover from p2 have |p| = p by auto
   ultimately have |k*p| > 2 by (auto simp only: abs-mult)
   moreover from k have ?Labp − ?Lap*?Lbp = k*p by auto
   ultimately have |?Labp − ?Lap*?Lbp| > 2 by auto
   moreover have ?Labp = 1 ∨ ?Labp = 0 ∨ ?Labp = −1
     by (simp add: Legendre-def)
   moreover have ?Lap*?Lbp = 1 ∨ ?Lap*?Lbp = 0 ∨ ?Lap*?Lbp = −1
     by (auto simp add: Legendre-def)
   ultimately show False by auto
  qed
  with k show ?thesis by auto
qed
```

Now show $(\frac{-3}{p}) = +1$ for primes $p \equiv 1 \bmod 6$.

```
lemma Legendre-1mod6: prime (6*m+1) ⟹ Legendre (−3) (6*m+1) = 1
proof −
  let ?p = 6*m+1
  let ?L = Legendre (−3) ?p
  let ?L1 = Legendre (−1) ?p
  let ?L3 = Legendre 3 ?p
  assume p: prime ?p
  from p have p': prime (nat ?p) by simp
  have neg1cube: (−1::int)^3 = −1 by simp
  have m1: m ≥ 1
  proof (rule ccontr)
    assume ¬ m ≥ 1 hence m ≤ 0 by simp
    with p show False by (auto simp add: prime-int-iff)
  qed
  hence pn3: ?p ≠ 3 and p2: ?p > 2 by auto
  with p have ?L = (Legendre (−1) ?p) * (Legendre 3 ?p)
    by (frule-tac a=−1 and b=3 in Legendre-zmult, auto)
  moreover have [Legendre (−1) ?p = (−1)^nat m] (mod ?p)
  proof −
    have nat((?p − 1) div 2) = (nat ?p − 1) div 2 by auto
    hence [?L1 = (−1)^(nat(((?p) − 1) div 2))] (mod ?p)
      using euler-criterion[of nat ?p −1] p' p2 by fastforce
    moreover have nat ((?p − 1) div 2) = 3* nat m
    proof −
      have (?p − 1) div 2 = 3*m by auto
      hence nat((?p − 1) div 2) = nat (3*m) by simp
      moreover have (3::int) ≥ 0 by simp
      ultimately show ?thesis by (simp add: nat-mult-distrib)
```

```
    qed
    moreover with neg1cube have (-1::int)^(3*nat m) = (-1)^nat m
      by (simp only: power-mult)
    ultimately show ?thesis by auto
  qed
  moreover have ?L3 = (-1)^nat m
  proof -
    have ?L3 * (Legendre ?p 3) = (-1)^nat m
    proof -
      have nat ((3 - 1) div 2 * ((6 * m + 1 - 1) div 2)) = 3*nat m by auto
      hence ?L3 * (Legendre ?p 3) = (-1::int) ^ (3*nat m)
        using Quadratic-Reciprocity-int[of 3 ?p] p' pn3 p2 by fastforce
      with neg1cube show ?thesis by (simp add: power-mult)
    qed
    moreover have Legendre ?p 3 = 1
    proof -
      have [1^2 = ?p] (mod 3) by (unfold cong-iff-dvd-diff dvd-def, auto)
      hence QuadRes 3 ?p by (unfold QuadRes-def, blast)
      moreover have ¬ [?p = 0] (mod 3)
      proof (rule ccontr, simp)
        assume [?p = 0] (mod 3)
        hence 3 dvd ?p by (simp add: cong-iff-dvd-diff)
        moreover have 3 dvd 6*m by (auto simp add: dvd-def)
        ultimately have 3 dvd ?p- 6*m by (simp only: dvd-diff)
        hence (3::int) dvd 1 by simp
        thus False by auto
      qed
      ultimately show ?thesis by (unfold Legendre-def, auto)
    qed
    ultimately show ?thesis by auto
  qed
  ultimately have [?L = (-1)^(nat m)*(-1)^(nat m)] (mod ?p)
    by (metis cong-scalar-right)
  hence [?L = (-1)^((nat m)+(nat m))] (mod ?p) by (simp only: power-add)
  moreover have (nat m)+(nat m) = 2*(nat m) by auto
  ultimately have [?L = (-1)^(2*(nat m))] (mod ?p) by simp
  hence [?L = ((-1)^2)^(nat m)] (mod ?p) by (simp only: power-mult)
  hence [1 = ?L] (mod ?p) by (auto simp add: cong-sym)
  hence ?p dvd 1 - ?L by (simp only: cong-iff-dvd-diff)
  moreover have ?L = -1 ∨ ?L = 0 ∨ ?L = 1 by (simp add: Legendre-def)
  ultimately have ?p dvd 2 ∨ ?p dvd 1 ∨ ?L = 1 by auto
  moreover
  { assume ?p dvd 2 ∨ ?p dvd 1
    with p2 have False by (auto simp add: zdvd-not-zless) }
  ultimately show ?thesis by auto
qed
```

Use this to prove that such primes can be written as $x^2 + 3y^2$.

```
lemma qf3-prime-exists: prime (6*m+1::int) ⟹ ∃ x y. 6*m+1 = x^2 + 3*y^2
proof -
  let ?p = 6*m+1
  assume p: prime ?p
```

```
  hence Legendre (−3) ?p = 1 by (rule Legendre-1mod6)
  moreover
  { assume ¬ QuadRes ?p (−3)
    hence Legendre (−3) ?p ≠ 1 by (unfold Legendre-def, auto) }
  ultimately have QuadRes ?p (−3) by auto
  then obtain s where s: [s^2 = −3] (mod ?p) by (auto simp add: QuadRes-def)
  hence ?p dvd s^2 − (−3::int) by (unfold cong-iff-dvd-diff, simp)
  moreover have s^2 −(−3::int) = s^2 + 3 by arith
  ultimately have ?p dvd s^2 + 3*1^2 by auto
  moreover have coprime s 1 by auto
  moreover have odd ?p
  proof −
    have ?p = 2*(3*m)+1 by simp
    thus ?thesis by simp
  qed
  moreover from p have prime ?p by simp
  ultimately have is-qfN ?p 3 using qf3-oddprimedivisor by blast
  thus ?thesis by (unfold is-qfN-def, auto)
qed

end

end
```

3 Fermat's last theorem, case $n = 3$

```
theory Fermat3
imports Quad-Form
begin

context
begin
```

Proof of Fermat's last theorem for the case $n = 3$:

$$\forall x, y, z : \; x^3 + y^3 = z^3 \Longrightarrow xyz = 0.$$

```
private lemma nat-relprime-power-divisors:
  assumes n0: 0 < n and abc: (a::nat)*b = c^n and relprime: coprime a b
  shows ∃ k. a = k^n
using assms proof (induct c arbitrary: a b rule: nat-less-induct)
case (1 c)
  show ?case
  proof (cases a > 1)
  case False
    hence a = 0 ∨ a = 1 by linarith
    thus ?thesis using n0 power-one zero-power by (simp only: eq-sym-conv) blast
  next
  case True
    then obtain p where p: prime p p dvd a using prime-factor-nat[of a] by blast
    hence h1: p dvd (c^n) using 1(3) dvd-mult2[of p a b] by presburger
```

```
    hence (p^n) dvd (c^n)
      using p(1) prime-dvd-power-nat[of p c n] dvd-power-same[of p c n] by blast
    moreover have h2: ¬ p dvd b
      using p ‹coprime a b› coprime-common-divisor-nat [of a b p] by auto
    hence ¬ (p^n) dvd b using n0 p(1)  dvd-power[of n p] gcd-nat.trans by blast
    ultimately have (p^n) dvd a
      using 1.prems p(1) prime-elem-divprod-pow [of p a b n] by simp
    then obtain a' c' where ac: a = p^n * a' c = p * c'
      using h1 dvdE[of p^n a] dvdE[of p c] prime-dvd-power-nat[of p c n] p(1) by meson
    hence p^n * (a' * b) = p^n * c'^n using 1(3)
      by (simp add: power-mult-distrib semiring-normalization-rules(18))
    hence a' * b = c'^n using p(1) by auto
    moreover have coprime a' b using 1(4) ac(1)
      by simp
    moreover have 0 < b 0 < a using h2 dvd-0-right gr0I True by fastforce+
   then have 0 < c 1 < p using p(1) 1(3) nat-0-less-mult-iff [of a b] n0 prime-gt-Suc-0-nat
      by simp-all
    hence c' < c using ac(2) by simp
    ultimately obtain k where a' = k^n using 1(1) n0 by presburger
    hence a = (p*k)^n using ac(1) by (simp add: power-mult-distrib)
    thus ?thesis by blast
  qed
qed

private lemma int-relprime-odd-power-divisors:
  assumes odd n and (a::int) * b = c ^ n and coprime a b
  shows ∃ k. a = k^n
proof −
  from assms have |a| * |b| = |c| ^ n
    by (simp add: abs-mult [symmetric] power-abs)
  then have nat |a| * nat |b| = nat |c| ^ n
    by (simp add: nat-mult-distrib [of |a| |b|, symmetric] nat-power-eq)
  moreover have coprime (nat |a|) (nat |b|) using assms(3) gcd-int-def by fastforce
  ultimately have ∃ k. nat |a| = k^n
    using nat-relprime-power-divisors[of n nat |a| nat |b| nat |c|] assms(1) by blast
  then obtain k' where k': nat |a| = k'^n by blast
  moreover define k where k = int k'
  ultimately have k: |a| = k^n using int-nat-eq[of |a|] of-nat-power[of k' n] by force
  { assume a ≠ k^n
    with k have a = −(k^n) by arith
    hence a = (−k)^n using assms(1) power-minus-odd by simp }
  thus ?thesis by blast
qed

private lemma factor-sum-cubes: (x::int)^3 + y^3 = (x+y)*(x^2 − x*y + y^2)
  by (simp add: eval-nat-numeral field-simps)

private lemma two-not-abs-cube: |x^3| = (2::int) ⟹ False
proof −
  assume |x^3| = 2
  hence x32: |x|^3 = 2 by (simp add: power-abs)
  have |x| ≥ 0 by simp
```

```
  moreover
  { assume |x| = 0 ∨ |x| = 1 ∨ |x| = 2
    with x32 have False by (auto simp add: power-0-left) }
  moreover
  { assume |x| > 2
    moreover have (0::int) ≤ 2 and (0::nat) < 3 by auto
    ultimately have |x|^3 > 2^3 by (simp only: power-strict-mono)
    with x32 have False by simp }
  ultimately show False by arith
qed
```

Shows there exists no solution $v^3 + w^3 = x^3$ with $vwx \neq 0$ and $coprime vw$ and x even, by constructing a solution with a smaller $|x^3|$.

```
private lemma no-rewritten-fermat3:
  ¬ (∃ v w. v^3+w^3 = x^3 ∧ v*w*x ≠ 0 ∧ even (x::int) ∧ coprime v w)
proof (induct x rule: infinite-descent0-measure[where V=λx. nat|x^3|])
  case (0 x) hence x^3 = 0 by arith
  hence x=0 by auto
  thus ?case by auto
next
  case (smaller x)
  then obtain v w where vwx:
    v^3+w^3=x^3 ∧ v*w*x ≠ 0 ∧ even x ∧ coprime v w (is ?P v w x)
    by auto
  then have coprime v w
    by simp
  have ∃ α β γ. ?P α β γ ∧ nat|γ^3| < nat|x^3|
  proof −
    — obtain coprime p and q such that v = p + q and w = p − q
    have vwOdd: odd v ∧ odd w
    proof (rule ccontr, case-tac odd v, simp-all)
      assume ve: even v
      hence even (v^3) by simp
      moreover from vwx have even (x^3) by simp
      ultimately have even (x^3−v^3) by simp
      moreover from vwx have x^3−v^3 = w^3 by simp
      ultimately have even (w^3) by simp
      hence even w by simp
      with ve have 2 dvd v ∧ 2 dvd w by auto
      hence 2 dvd gcd v w by simp
      with vwx show False by simp
    next
      assume odd v and even w
      hence odd (v^3) and even (w^3)
        by auto
      hence odd (w^3 + v^3) by simp
      with vwx have odd (x^3) by (simp add: add.commute)
      hence odd x by simp
      with vwx show False by auto
    qed
    hence even (v+w) ∧ even (v−w) by simp
    then obtain p q where pq: v+w = 2*p ∧ v−w = 2*q
```

```
  using evenE[of v+w] evenE[of v−w] by meson
hence vw: v = p+q ∧ w = p−q by auto
— show that x^3 = (2p)(p^2 + 3q^2) and that these factors are
— either coprime (first case), or have 3 as g.c.d. (second case)
have vwpq: v^3 + w^3 = (2*p)*(p^2 + 3*q^2)
proof −
  have 2*(v^3 + w^3) = 2*(v+w)*(v^2 − v*w + w^2)
    by (simp only: factor-sum-cubes)
  also from pq have ... = 4*p*(v^2 − v*w + w^2) by auto
  also have ... = p*((v+w)^2 + 3*(v−w)^2)
    by (simp add: eval-nat-numeral field-simps)
  also with pq have ... = p*((2*p)^2 + 3*(2*q)^2) by simp
  also have ... = 2*(2*p)*(p^2+3*q^2) by (simp add: power-mult-distrib)
  finally show ?thesis by simp
qed
let ?g = gcd (2 * p) (p^2 + 3 * q^2)
have g1: ?g ≥ 1
proof (rule ccontr)
  assume ¬ ?g ≥ 1
  then have ?g < 0 ∨ ?g = 0 unfolding not-le by arith
  moreover have ?g ≥ 0 by simp
  ultimately have ?g = 0 by arith
  hence p = 0 by simp
  with vwpq vwx ‹0 < nat|x^3|› show False by auto
qed
have gOdd: odd ?g
proof (rule ccontr)
  assume ¬ odd ?g
  hence2 dvd p^2+3*q^2 by simp
  then obtain k where k: p^2 + 3*q^2 = 2*k by (auto simp add: dvd-def)
  hence 2*(k − 2*q^2) = p^2−q^2 by auto
  also have ... = (p+q)*(p−q) by (simp add: power2-eq-square algebra-simps)
  finally have v*w = 2*(k − 2*q^2) using vw by presburger
  hence even (v*w) by auto
  hence even (v) ∨ even (w) by simp
  with vwOdd show False by simp
qed
then have even-odd-p-q: even p ∧ odd q ∨ odd p ∧ even q
  by auto
— first case: p is not a multiple of 3; hence 2p and p^2 + 3q^2
— are coprime; hence both are cubes
{ assume p3: ¬ 3 dvd p
  have g3: ¬ 3 dvd ?g
  proof (rule ccontr)
    assume ¬ ¬ 3 dvd ?g hence 3 dvd 2*p by simp
    hence (3::int) dvd 2 ∨ 3 dvd p
      using prime-dvd-multD[of 3] by (fastforce simp add: prime-dvd-mult-iff)
    with p3 show False by arith
  qed
  from ‹coprime v w› have pq-relprime: coprime p q
  proof (rule coprime-imp-coprime)
    fix c
```

```
  assume c dvd p and c dvd q
  then have c dvd p + q and c dvd p − q
    by simp-all
  with vw show c dvd v and c dvd w
    by simp-all
qed
from ‹coprime p q› have coprime p (q^2)
  by simp
then have factors-relprime: coprime (2 ∗ p) (p^2 + 3 ∗ q^2)
proof (rule coprime-imp-coprime)
  fix c
  assume g2p: c dvd 2 ∗ p and gpq: c dvd p^2 + 3 ∗ q^2
  have coprime 2 c
    using g2p gpq even-odd-p-q dvd-trans [of 2 c p^2 + 3 ∗ q^2]
    by auto
  with g2p show c dvd p
    by (simp add: coprime-dvd-mult-left-iff ac-simps)
  then have c dvd p^2
    by (simp add: power2-eq-square)
  with gpq have c dvd 3 ∗ q^2
    by (simp add: dvd-add-right-iff)
  moreover have coprime 3 c
    using ‹c dvd p› p3 dvd-trans [of 3 c p]
    by (auto intro: prime-imp-coprime)
  ultimately show c dvd q^2
    by (simp add: coprime-dvd-mult-right-iff ac-simps)
qed
moreover from vwx vwpq have pqx: (2∗p)∗(p^2 + 3∗q^2) = x^3 by auto
ultimately have ∃ c. 2∗p = c^3 by (simp add: int-relprime-odd-power-divisors)
then obtain c where c: c^3 = 2∗p by auto
from pqx factors-relprime have coprime (p^2 + 3∗q^2) (2∗p)
  and (p^2 + 3∗q^2)∗(2∗p) = x^3 by (auto simp add: ac-simps)
hence ∃ d. p^2 + 3∗q^2 = d^3 by (simp add: int-relprime-odd-power-divisors)
then obtain d where d: p^2 + 3∗q^2 = d^3 by auto
have odd d
proof (rule ccontr)
  assume ¬ odd d
  hence even (d^3) by simp
  hence 2 dvd d^3 by simp
  moreover have 2 dvd 2∗p by (rule dvd-triv-left)
  ultimately have 2 dvd gcd (2∗p) (d^3) by simp
  with d factors-relprime show False by simp
qed
with d pq-relprime have coprime p q ∧ p^2 + 3∗q^2 = d^3 ∧ odd d
  by simp
hence is-cube-form p q by (rule qf3-cube-impl-cube-form)
then obtain a b where p = a^3 − 9∗a∗b^2 ∧ q = 3∗a^2∗b − 3∗b^3
  by (unfold is-cube-form-def, auto)
hence ab: p = a∗(a+3∗b)∗(a− 3∗b) ∧ q = b∗(a+b)∗(a−b)∗3
  by (simp add: eval-nat-numeral field-simps)
with c have abc: (2∗a)∗(a+3∗b)∗(a− 3∗b) = c^3 by auto
from pq-relprime ab have ab-relprime: coprime a b
```

```
    by (auto intro: coprime-imp-coprime)
  then have ab1: coprime (2 ∗ a) (a + 3 ∗ b)
  proof (rule coprime-imp-coprime)
    fix h
    assume h2a: h dvd 2 ∗ a and hab: h dvd a + 3 ∗ b
    have coprime 2 h
      using ab even-odd-p-q hab dvd-trans [of 2 h a + 3 ∗ b]
      by auto
    with h2a show h dvd a
      by (simp add: coprime-dvd-mult-left-iff ac-simps)
    with hab have h dvd 3 ∗ b and ¬ 3 dvd h
      using dvd-trans [of 3 h a] ab ‹¬ 3 dvd p›
      by (auto simp add: dvd-add-right-iff)
    moreover have coprime 3 h
      using ‹¬ 3 dvd h› by (auto intro: prime-imp-coprime)
    ultimately show h dvd b
      by (simp add: coprime-dvd-mult-left-iff ac-simps)
  qed
  then have [simp]: even b ⟷ odd a
    and ab3: coprime a (a + 3 ∗ b)
    by simp-all
  from ‹coprime a b› have ab4: coprime a (a − 3 ∗ b)
  proof (rule coprime-imp-coprime)
    fix h
    assume h2a: h dvd a and hab: h dvd a − 3 ∗ b
    then show h dvd a
      by simp
    with hab have h dvd 3 ∗ b and ¬ 3 dvd h
      using dvd-trans [of 3 h a] ab ‹¬ 3 dvd p› dvd-add-right-iff [of h a − 3 ∗ b]
      by auto
    moreover have coprime 3 h
      using ‹¬ 3 dvd h› by (auto intro: prime-imp-coprime)
    ultimately show h dvd b
      by (simp add: coprime-dvd-mult-left-iff ac-simps)
  qed
  from ab1 have ab2: coprime (a + 3 ∗ b) (a − 3 ∗ b)
    by (rule coprime-imp-coprime)
      (use dvd-add [of - a + 3 ∗ b a − 3 ∗ b] in simp-all)
  have ∃ k l m. 2 ∗ a = k ^ 3 ∧ a + 3 ∗ b = l ^ 3 ∧ a − 3 ∗ b = m ^ 3
    using ab2 ab3 ab4 abc
      int-relprime-odd-power-divisors [of 3 2 ∗ a (a + 3 ∗ b) ∗ (a − 3 ∗ b) c]
      int-relprime-odd-power-divisors [of 3 (a + 3 ∗ b) 2 ∗ a ∗ (a − 3 ∗ b) c]
      int-relprime-odd-power-divisors [of 3 (a − 3 ∗ b) 2 ∗ a ∗ (a + 3 ∗ b) c]
    by auto (auto simp add: ac-simps)
  then obtain α β γ where albega:
    2∗a = γ^3 ∧ a − 3∗b = α^3 ∧ a+3∗b = β^3 by auto
  — show this is a (smaller) solution
  hence α^3 + β^3 = γ^3 by auto
  moreover have α∗β∗γ ≠ 0
  proof (rule ccontr, safe)
    assume α ∗ β ∗ γ = 0
    with albega ab have p=0 by (auto simp add: power-0-left)
```

```
    with vwpq vwx show False by auto
  qed
  moreover have even γ
  proof −
    have even (2∗a) by simp
    with albega have even (γˆ3) by simp
    thus ?thesis by simp
  qed
  moreover have coprime α β
  using ab2 proof (rule coprime-imp-coprime)
    fix h
    assume ha: h dvd α and hb: h dvd β
    then have h dvd α ∗ αˆ2 ∧ h dvd β ∗ βˆ2 by simp
    then have h dvd αˆSuc 2 ∧ h dvd βˆSuc 2 by (auto simp only: power-Suc)
    with albega show h dvd a − 3 ∗ b h dvd a + 3 ∗ b by auto
  qed
  moreover have nat|γˆ3| < nat|xˆ3|
  proof −
    let ?A = pˆ2 + 3∗qˆ2
    from vwx vwpq have xˆ3 = 2∗p∗?A by auto
    also with ab have . . . = 2∗a∗((a+3∗b)∗(a− 3∗b)∗?A) by auto
    also with albega have . . . = γˆ3 ∗((a+3∗b)∗(a− 3∗b)∗?A) by auto
    finally have eq: |xˆ3| = |γˆ3| ∗ |(a+3∗b)∗(a− 3∗b)∗?A|
      by (auto simp add: abs-mult)
    with ‹0 < nat|xˆ3|› have |(a+3∗b)∗(a− 3∗b)∗?A| > 0 by auto
    hence eqpos: |(a+3∗b)∗(a− 3∗b)| > 0 by auto
    moreover have Ag1: |?A| > 1
    proof −
      have Aqf3: is-qfN ?A 3 by (auto simp add: is-qfN-def)
      moreover have triv3b: (3::int) ≥ 1 by simp
      ultimately have ?A ≥ 0 by (simp only: qfN-pos)
      hence ?A > 1 ∨ ?A = 0 ∨ ?A =1 by arith
      moreover
      { assume ?A = 0 with triv3b have p = 0 ∧ q = 0 by (rule qfN-zero)
        with vwpq vwx have False by auto }
      moreover
      { assume A1: ?A = 1
        have q=0
        proof (rule ccontr)
          assume q ≠ 0
          hence qˆ2 > 0 by simp
          hence 3∗qˆ2 > 1 by arith
          moreover have pˆ2 ≥ 0 by (rule zero-le-power2)
          ultimately have ?A > 1 by arith
          with A1 show False by simp
        qed
        with pq-relprime have |p| = 1 by simp
        with vwpq vwx A1 have |xˆ3| = 2 by auto
        hence False by (rule two-not-abs-cube) }
      ultimately show ?thesis by auto
    qed
    ultimately have
```

```
      |(a+3*b)*(a- 3*b)|*1 < |(a+3*b)*(a- 3*b)|*|?A|
      by (simp only: zmult-zless-mono2)
    with eqpos have |(a+3*b)*(a- 3*b)|*|?A| > 1 by arith
    hence |(a+3*b)*(a- 3*b)*?A| > 1 by (auto simp add: abs-mult)
    moreover have |γ^3| > 0
    proof -
      from eq have |γ^3| = 0 ⟹ |x^3|=0 by auto
      with ‹0 < nat|x^3|› show ?thesis by auto
    qed
    ultimately have |γ^3| * 1 < |γ^3| * |(a+3*b)*(a- 3*b)*?A|
      by (rule zmult-zless-mono2)
    with eq have |x^3| > |γ^3| by auto
    thus ?thesis by arith
  qed
  ultimately have ?thesis by auto }
moreover
— second case: p = 3r and hence x^3 = (18r)(q^2 + 3r^2) and these
— factors are coprime; hence both are cubes
{ assume p3: 3 dvd p
  then obtain r where r: p = 3*r by (auto simp add: dvd-def)
  moreover have 3 dvd 3*(3*r^2 + q^2) by (rule dvd-triv-left)
  ultimately have pq3: 3 dvd p^2+3*q^2 by (simp add: power-mult-distrib)
  moreover from p3 have 3 dvd 2*p by (rule dvd-mult)
  ultimately have g3: 3 dvd ?g by simp
  from ‹coprime v w› have qr-relprime: coprime q r
  proof (rule coprime-imp-coprime)
    fix h
    assume hq: h dvd q h dvd r
    with r have h dvd p by simp
    with hq have h dvd p + q h dvd p - q
      by simp-all
    with vw show h dvd v h dvd w
      by simp-all
  qed
  have factors-relprime: coprime (18*r) (q^2 + 3*r^2)
  proof -
    from g3 obtain k where k: ?g = 3*k by (auto simp add: dvd-def)
    have k = 1
    proof (rule ccontr)
      assume k ≠ 1
      with g1 k have k > 1 by auto
      then obtain h where h: prime h ∧ h dvd k
        using prime-divisor-exists[of k] by auto
      with k have hg: 3*h dvd ?g by (auto simp add: mult-dvd-mono)
      hence 3*h dvd p^2 + 3*q^2 and hp: 3*h dvd 2*p by auto
      then obtain s where s: p^2 + 3*q^2 = (3*h)*s
        by (auto simp add: dvd-def)
      with r have rqh: 3*r^2+q^2 = h*s by (simp add: power-mult-distrib)
      from hp r have 3*h dvd 3*(2*r) by simp
      moreover have (3::int) ≠ 0 by simp
      ultimately have h dvd 2*r by (rule zdvd-mult-cancel)
      with h have h dvd 2 ∨ h dvd r
```

```
        by (auto dest: prime-dvd-multD)
      moreover have ¬ h dvd 2
      proof (rule ccontr, simp)
        assume h dvd 2
        with h have h=2 using zdvd-not-zless[of 2 h] by (auto simp: prime-int-iff)
        with hg have 2*3 dvd ?g by auto
        hence 2 dvd ?g by (rule dvd-mult-left)
        with gOdd show False by simp
      qed
      ultimately have hr: h dvd r by simp
      then obtain t where r = h*t by (auto simp add: dvd-def)
      hence t: r^2 = h*(h*t^2) by (auto simp add: power2-eq-square)
      with rqh have h*s = h*(3*h*t^2) + q^2 by simp
      hence q^2 = h*(s − 3*h*t^2) by (simp add: right-diff-distrib)
      hence h dvd q^2 by simp
      with h have h dvd q using prime-dvd-multD[of h q q]
        by (simp add: power2-eq-square)
      with hr have h dvd gcd q r by simp
      with h qr-relprime show False by (unfold prime-def, auto)
    qed
    with k r have 3 = gcd (2*(3*r)) ((3*r)^2 + 3*q^2) by auto
    also have ... = gcd (3*(2*r)) (3*(3*r^2 + q^2))
      by (simp add: power-mult-distrib)
   also have ... = 3 * gcd (2*r) (3*r^2 + q^2) using gcd-mult-distrib-int[of 3] by
auto
    finally have coprime (2*r) (3*r^2 + q^2)
      by (auto dest: gcd-eq-1-imp-coprime)
    moreover have coprime 9 (3*r^2 + q^2)
    using ‹coprime v w› proof (rule coprime-imp-coprime)
      fix h :: int
      assume ¬ is-unit h
      assume h9: h dvd 9 and hrq: h dvd 3 * r² + q²
      have prime (3::int)
        by simp
      moreover from ‹h dvd 9› have h dvd 3²
        by simp
      ultimately obtain k where normalize h = 3 ^ k
        by (rule divides-primepow)
      with ‹¬ is-unit h› have 0 < k
        by simp
      with ‹normalize h = 3 ^ k› have |h| = 3 * 3 ^ (k − 1)
        by (cases k) simp-all
      then have 3 dvd |h| ..
      then have 3 dvd h
        by simp
      then have 3 dvd 3 * r² + q²
        using hrq by (rule dvd-trans)
      then have 3 dvd q²
        by presburger
      then have 3 dvd q
        using prime-dvd-power-int [of 3 q 2] by auto
      with p3 have 3 dvd p + q and 3 dvd p − q
```

```
      by simp-all
    with vw have 3 dvd v and 3 dvd w
      by simp-all
    with ‹coprime v w› have is-unit (3::int)
      by (rule coprime-common-divisor)
    then show h dvd v and h dvd w
      by simp-all
  qed
  ultimately have coprime (2 * r * 9) (3 * r² + q²)
    by (simp only: coprime-mult-left-iff)
  then show ?thesis
    by (simp add: ac-simps)
qed
moreover have rqx: (18*r)*(q^2 + 3*r^2) = x^3
proof −
  from vwx vwpq have x^3 = 2*p*(p^2 + 3*q^2) by auto
  also with r have ... = 2*(3*r)*(9*r^2 + 3*q^2)
    by (auto simp add: power2-eq-square)
  finally show ?thesis by auto
qed
ultimately have ∃ c. 18*r = c^3
  by (simp add: int-relprime-odd-power-divisors)
then obtain c1 where c1: c1^3 = 3*(6*r) by auto
hence 3 dvd c1^3 and prime (3::int) by auto
hence 3 dvd c1 using prime-dvd-power[of 3] by fastforce
with c1 obtain c where c: 3*c^3 = 2*r
  by (auto simp add: power-mult-distrib dvd-def)
from rqx factors-relprime have coprime (q^2 + 3*r^2) (18*r)
  and (q^2 + 3*r^2)*(18*r) = x^3 by (auto simp add: ac-simps)
hence ∃ d. q^2 + 3*r^2 = d^3
  by (simp add: int-relprime-odd-power-divisors)
then obtain d where d: q^2 + 3*r^2 = d^3 by auto
have odd d
proof (rule ccontr)
  assume ¬ odd d
  hence 2 dvd d^3 by simp
  moreover have 2 dvd 2*(9*r) by (rule dvd-triv-left)
  ultimately have 2 dvd gcd (2*(9*r)) (d^3) by simp
  with d factors-relprime show False by auto
qed
with d qr-relprime have coprime q r ∧ q^2 + 3*r^2 = d^3 ∧ odd d
  by simp
hence is-cube-form q r by (rule qf3-cube-impl-cube-form)
then obtain a b where q = a^3 − 9*a*b^2 ∧ r = 3*a^2*b − 3*b^3
  by (unfold is-cube-form-def, auto)
hence ab: q = a*(a+3*b)*(a− 3*b) ∧ r = b*(a+b)*(a−b)*3
  by (simp add: eval-nat-numeral field-simps)
with c have abc: (2*b)*(a+b)*(a−b) = c^3 by auto
from qr-relprime ab have ab-relprime: coprime a b
  by (auto intro: coprime-imp-coprime)
then have ab1: coprime (2*b) (a+b)
proof (rule coprime-imp-coprime)
```

```
    fix h
    assume h2b: h dvd 2*b and hab: h dvd a+b
    have odd h
    proof
      assume even h
      then have even (a + b)
        using hab by (rule dvd-trans)
      then have even (a+3*b)
        by simp
      with ab have even q even r
        by auto
      then show False
        using coprime-common-divisor-int qr-relprime by fastforce
    qed
    with h2b show h dvd b
      using coprime-dvd-mult-right-iff [of h 2 b] by simp
    with hab show h dvd a
      using dvd-diff [of h a + b b] by simp
  qed
  from ab1 have ab2: coprime (a+b) (a−b)
  proof (rule coprime-imp-coprime)
    fix h
    assume hab1: h dvd a+b and hab2: h dvd a−b
    then show h dvd 2*b using dvd-diff[of h a+b a−b] by fastforce
  qed
  from ab1 have ab3: coprime (a−b) (2*b)
  proof (rule coprime-imp-coprime)
    fix h
    assume hab: h dvd a−b and h2b: h dvd 2*b
    have a−b+2*b = a+b by simp
    then show h dvd a+b using hab h2b dvd-add [of h a−b 2*b] by presburger
  qed
  then have [simp]: even b ⟷ odd a
    by simp
  have ∃ k l m. 2*b = k^3 ∧ a+b = l^3 ∧ a−b = m^3
    using abc ab1 ab2 ab3
      int-relprime-odd-power-divisors [of 3 2 * b (a + b) * (a − b) c]
      int-relprime-odd-power-divisors [of 3 a + b (2 * b) * (a − b) c]
      int-relprime-odd-power-divisors [of 3 a − b (2 * b) * (a + b) c]
    by simp (simp add: ac-simps, simp add: algebra-simps)
  then obtain α1 β γ where a1: 2*b = γ^3 ∧ a−b = α1^3 ∧ a+b = β^3
    by auto
  then obtain α where α = −α1 by auto
  — show this is a (smaller) solution
  with a1 have a2: α^3 = b−a by auto
  with a1 have α^3 + β^3 = γ^3 by auto
  moreover have α*β*γ ≠ 0
  proof (rule ccontr, safe)
    assume α * β * γ = 0
    with a1 a2 ab have r=0 by (auto simp add: power-0-left)
    with r vwpq vwx show False by auto
  qed
```

```
    moreover have even γ
    proof -
      have even (2*b) by simp
      with a1 have even (γ^3) by simp
      thus ?thesis by simp
    qed
    moreover have coprime α β
    using ab2 proof (rule coprime-imp-coprime)
      fix h
      assume ha: h dvd α and hb: h dvd β
      then have h dvd α * α^2 and h dvd β * β^2 by simp-all
      then have h dvd α^Suc 2 and h dvd β^Suc 2 by (auto simp only: power-Suc)
      with a1 a2 have h dvd b - a and h dvd a + b by auto
      then show h dvd a + b and h dvd a - b
        by (simp-all add: dvd-diff-commute)
    qed
    moreover have nat|γ^3| < nat|x^3|
    proof -
      let ?A = p^2 + 3*q^2
      from vwx vwpq have x^3 = 2*p*?A by auto
      also with r have ... = 6*r*?A by auto
      also with ab have ... = 2*b*(9*(a+b)*(a-b)*?A) by auto
      also with a1 have ... = γ^3 *(9*(a+b)*(a-b)*?A) by auto
      finally have eq: |x^3| = |γ^3| * |9*(a+b)*(a-b)*?A|
        by (auto simp add: abs-mult)
      with ‹0 < nat|x^3|› have |9*(a+b)*(a-b)*?A| > 0 by auto
      hence |(a+b)*(a-b)*?A| ≥ 1 by arith
      hence |9*(a+b)*(a-b)*?A| > 1 by arith
      moreover have |γ^3| > 0
      proof -
        from eq have |γ^3| = 0 ⟹ |x^3|=0 by auto
        with ‹0 < nat|x^3|› show ?thesis by auto
      qed
      ultimately have |γ^3| * 1 < |γ^3| * |9*(a+b)*(a-b)*?A|
        by (rule zmult-zless-mono2)
      with eq have |x^3| > |γ^3| by auto
      thus ?thesis by arith
    qed
    ultimately have ?thesis by auto }
  ultimately show ?thesis by auto
 qed
 thus ?case by auto
qed
```

The theorem. Puts equation in requested shape.

```
theorem fermat-3:
  assumes ass: (x::int)^3 + y^3 = z^3
  shows x*y*z=0
proof (rule ccontr)
  let ?g = gcd x y
  let ?c = z div ?g
  assume xyz0: x*y*z≠0
```

```
— divide out the g.c.d.
hence x ≠ 0 ∨ y ≠ 0 by simp
then obtain a b where ab: x = ?g*a ∧ y = ?g*b ∧ coprime a b
  using gcd-coprime-exists[of x y] by (auto simp: mult.commute)
moreover have abc: ?c*?g = z ∧ a^3 + b^3 = ?c^3 ∧ a*b*?c ≠ 0
proof −
  from xyz0 have g0: ?g≠0 by simp
  have zgab: z^3 = ?g^3 * (a^3+b^3)
  proof −
    from ab and ass have z^3 = (?g*a)^3+(?g*b)^3 by simp
    thus ?thesis by (simp only: power-mult-distrib distrib-left)
  qed
  have cgz: ?c * ?g = z
  proof −
    from zgab have ?g^3 dvd z^3 by simp
    hence ?g dvd z by simp
    thus ?thesis by (simp only: ac-simps dvd-mult-div-cancel)
  qed
  moreover have a^3 + b^3 = ?c^3
  proof −
    have ?c^3 * ?g^3 = (a^3+b^3)*?g^3
    proof −
      have ?c^3 * ?g^3 = (?c*?g)^3 by (simp only: power-mult-distrib)
      also with cgz have ... = z^3 by simp
      also with zgab have ... = ?g^3*(a^3+b^3) by simp
      finally show ?thesis by simp
    qed
    with g0 show ?thesis by auto
  qed
  moreover from ab and xyz0 and cgz have a*b*?c≠0 by auto
  ultimately show ?thesis by simp
qed
— make both sides even
from ab have coprime (a ^ 3) (b ^ 3)
  by simp
have ∃ u v w. u^3 + v^3 = w^3 ∧ u*v*w≠(0::int) ∧ even w ∧ coprime u v
proof −
  let ?Q u v w = u^3 + v^3 = w^3 ∧ u*v*w≠(0::int) ∧ even w ∧ coprime u v
  have even a ∨ even b ∨ even ?c
  proof (rule ccontr)
    assume ¬(even a ∨ even b ∨ even ?c)
    hence aodd: odd a and odd b ∧ odd ?c by auto
    hence even (?c^3 − b^3) by simp
    moreover from abc have ?c^3−b^3 = a^3 by simp
    ultimately have even (a^3) by auto
    hence even (a) by simp
    with aodd show False by simp
  qed
  moreover
  { assume even (a)
    then obtain u v w where uvwabc: u = −b ∧ v = ?c ∧ w = a ∧ even w
      by auto
```

```
    moreover with abc have u*v*w≠0 by auto
    moreover have uvw: u^3+v^3=w^3
    proof -
      from uvwabc have u^3 + v^3 = (-1*b)^3 + ?c^3 by simp
      also have ... = (-1)^3*b^3 + ?c^3 by (simp only: power-mult-distrib)
      also have ... = - (b^3) + ?c^3 by auto
      also with abc and uvwabc have ... = w^3 by auto
      finally show ?thesis by simp
    qed
    moreover have coprime u v
    using ‹coprime (a ^ 3) (b ^ 3)› proof (rule coprime-imp-coprime)
      fix h
      assume hu: h dvd u and h dvd v
      with uvwabc have h dvd ?c*?c^2 by (simp only: dvd-mult2)
      with abc have h dvd a^3+b^3 using power-Suc[of ?c 2] by simp
      moreover from hu uvwabc have hb3: h dvd b*b^2 by simp
      ultimately have h dvd a^3+b^3-b^3
        using power-Suc [of b 2] dvd-diff [of h a ^ 3 + b ^ 3 b ^ 3] by simp
      with hb3 show h dvd a^3 h dvd b^3 using power-Suc[of b 2] by auto
    qed
    ultimately have ?Q u v w using ‹even a› by simp
    hence ?thesis by auto }
  moreover
  { assume even b
    then obtain u v w where uvwabc: u = -a ∧ v = ?c ∧ w = b ∧ even w
      by auto
    moreover with abc have u*v*w≠0 by auto
    moreover have uvw: u^3+v^3=w^3
    proof -
      from uvwabc have u^3 + v^3 = (-1*a)^3 + ?c^3 by simp
      also have ... = (-1)^3*a^3 + ?c^3 by (simp only: power-mult-distrib)
      also have ... = - (a^3) + ?c^3 by auto
      also with abc and uvwabc have ... = w^3 by auto
      finally show ?thesis by simp
    qed
    moreover have coprime u v
    using ‹coprime (a ^ 3) (b ^ 3)› proof (rule coprime-imp-coprime)
      fix h
      assume hu: h dvd u and h dvd v
      with uvwabc have h dvd ?c*?c^2 by (simp only: dvd-mult2)
      with abc have h dvd a^3+b^3 using power-Suc[of ?c 2] by simp
      moreover from hu uvwabc have hb3: h dvd a*a^2 by simp
      ultimately have h dvd a^3+b^3-a^3
        using power-Suc [of a 2] dvd-diff [of h a ^ 3 + b ^ 3 a ^ 3] by simp
      with hb3 show h dvd a^3 and h dvd b^3 using power-Suc[of a 2] by auto
    qed
    ultimately have ?Q u v w using ‹even b› by simp
    hence ?thesis by auto }
  moreover
  { assume even ?c
    then obtain u v w where uvwabc: u = a ∧ v = b ∧ w = ?c ∧ even w
      by auto
```

```
    with abc ab have ?thesis by auto }
  ultimately show ?thesis by auto
 qed
 hence ∃ w. ∃ u v. u^3 + v^3 = w^3 ∧ u*v*w ≠ (0::int) ∧ even w ∧ coprime u v
  by auto
 — show contradiction using the earlier result
 thus False by (auto simp only: no-rewritten-fermat3)
qed

corollary fermat-mult3:
 assumes xyz: (x::int)^n + y^n = z^n and n: 3 dvd n
 shows x*y*z=0
proof −
 from n obtain m where n = m*3 by (auto simp only: ac-simps dvd-def)
 with xyz have (x^m)^3 + (y^m)^3 = (z^m)^3 by (simp only: power-mult)
 hence (x^m)*(y^m)*(z^m) = 0 by (rule fermat-3)
 thus ?thesis by auto
qed

end

end
```

References

[DM05] David Delahaye and Micaela Mayero. Diophantus' 20th problem and fermat's last theorem for n=4: Formalization of fermat's proofs in the coq proof assistant. http://hal.archives-ouvertes.fr/hal-00009425/en/, 2005.

[Edw77] Harold M. Edwards. *Fermat's Last Theorem. A Genetic Introduction to Algebraic Number Theory*. Springer Verlag, 1977.

[Oos07] Roelof Oosterhuis. Mechanised theorem proving: Exponents 3 and 4 of Fermat's Last Theorem in Isabelle. Master's thesis, University of Groningen, 2007. http://www.roelofoosterhuis.nl/MScthesis.pdf.

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