

Probabilistic Primality Testing

Daniel Stüwe and Manuel Eberl

May 26, 2024

Abstract

The most efficient known primality tests are *probabilistic* in the sense that they use randomness and may, with some probability, mistakenly classify a composite number as prime – but never a prime number as composite. Examples of this are the Miller–Rabin test, the Solovay–Strassen test, and (in most cases) Fermat’s test.

This entry defines these three tests and proves their correctness. It also develops some of the number-theoretic foundations, such as Carmichael numbers and the Jacobi symbol with an efficient executable algorithm to compute it.

Contents

| | | |
|-----------|-----------------------------------------------------------|-----------|
| 1 | Additional Facts about the Legendre Symbol | 3 |
| 2 | Auxiliary Material | 7 |
| 3 | The Jacobi Symbol | 17 |
| 4 | Residue Rings of Natural Numbers | 29 |
| 4.1 | The multiplicative group of residues modulo n | 29 |
| 4.2 | The ring of residues modulo n | 30 |
| 4.3 | The ring of residues modulo a prime | 32 |
| 4.4 | -1 in residue rings | 34 |
| 5 | Additional Material on Quadratic Residues | 35 |
| 6 | Euler Witnesses | 38 |
| 7 | Carmichael Numbers | 49 |
| 8 | Fermat Witnesses | 58 |
| 9 | A Generic View on Probabilistic Prime Tests | 74 |
| 10 | Fermat's Test | 76 |
| 11 | The Miller–Rabin Test | 78 |
| 12 | The Solovay–Strassen Test | 78 |

1 Additional Facts about the Legendre Symbol

theory *Legendre-Symbol*

imports

HOL-Number-Theory.Number-Theory

begin

lemma *basic-cong[simp]*:

fixes $p :: int$

assumes $2 < p$

shows $[-1 \neq 1] (mod\ p)$

$[1 \neq -1] (mod\ p)$

$[0 \neq 1] (mod\ p)$

$[1 \neq 0] (mod\ p)$

$[0 \neq -1] (mod\ p)$

$[-1 \neq 0] (mod\ p)$

using *assms* **by** (*simp-all add: cong-iff-dvd-diff zdvd-not-zless*)

lemma *[simp]*: $0 < n \implies (a\ mod\ 2) ^ n = a\ mod\ 2$ **for** $n :: nat$ **and** $a :: int$

by (*metis not-mod-2-eq-0-eq-1 power-one zero-power*)

lemma *Legendre-in-cong-eq*:

fixes $p :: int$

assumes $p > 2$ **and** $b \in \{-1, 0, 1\}$

shows $[Legendre\ a\ m = b] (mod\ p) \longleftrightarrow Legendre\ a\ m = b$

using *assms* **unfolding** *Legendre-def* **by** *auto*

lemma *Legendre-p-eq-2[simp]*: $Legendre\ a\ 2 = a\ mod\ 2$

by (*clarsimp simp: Legendre-def QuadRes-def cong-iff-dvd-diff*) *presburger*

lemma *Legendre-p-eq-1[simp]*: $Legendre\ a\ 1 = 0$ **by** (*simp add: Legendre-def*)

lemma *euler-criterion-int*:

assumes *prime* p **and** $2 < p$

shows $[Legendre\ a\ p = a^{(nat\ p-1)\ div\ 2}] (mod\ p)$

using *euler-criterion assms prime-int-nat-transfer*

by (*metis int-nat-eq nat-numeral prime-gt-0-int zless-nat-conj*)

lemma *QuadRes-neg[simp]*: $QuadRes\ (-p)\ a = QuadRes\ p\ a$ **unfolding** *QuadRes-def*

by *auto*

lemma *Legendre-neg[simp]*: $Legendre\ a\ (-p) = Legendre\ a\ p$ **unfolding** *Legendre-def*

by *auto*

lemma *Legendre-mult[simp]*:

assumes *prime* p

shows $Legendre\ (a*b)\ p = Legendre\ a\ p * Legendre\ b\ p$

proof –

consider $p = 2 \mid p > 2$

```

using assms order-le-less prime-ge-2-int by auto

thus ?thesis proof (cases)
case 1
  then show ?thesis
    by (metis Legendre-p-eq-2 mod-mult-eq mod-self mult-cancel-right2
        mult-eq-0-iff not-mod-2-eq-1-eq-0 one-mod-two-eq-one)
next
case 2
  hence [Legendre (a*b) p = (a*b) ^((nat p-1) div 2)] (mod p)
    using euler-criterion-int assms by blast

  also have [(a*b) ^((nat p-1) div 2) = a ^((nat p-1) div 2) * b ^((nat p-1) div 2)] (mod p)
    by (simp add: field-simps)

  also have [a ^((nat p-1) div 2) * b ^((nat p-1) div 2) = Legendre a p * Legendre b p] (mod p)
    using cong-sym[OF euler-criterion-int] assms 2 cong-mult by blast

  finally show ?thesis using Legendre-in-cong-eq[OF 2] by (simp add: Legendre-def)
qed
qed

lemma QuadRes-mod[simp]: p dvd n ==> QuadRes p (a mod n) = QuadRes p a
  by (simp add: mod-mod-cancel QuadRes-def cong-def)

lemma Legendre-mod[simp]: p dvd n ==> Legendre (a mod n) p = Legendre a p
  by (simp add: mod-mod-cancel Legendre-def cong-def)

lemma two-cong-0-iff: [2 = 0] (mod p) <=> p = 1 ∨ p = 2 for p :: nat
  unfolding cong-altdef-nat[of 0 2 p, simplified]
  using dvd-refl prime-nat-iff two-is-prime-nat by blast

lemma two-cong-0-iff-nat: [2 = 0] (mod int p) <=> p = 1 ∨ p = 2
  unfolding cong-iff-dvd-diff
  using two-is-prime-nat prime-nat-iff int-dvd-int-iff[of p 2]
  by auto

lemma two-cong-0-iff-int: p > 0 ==> [2 = 0] (mod p) <=> p = 1 ∨ p = 2 for p :: int
  by (metis of-nat-numeral pos-int-cases semiring-char-0-class.of-nat-eq-1-iff two-cong-0-iff-nat)

lemma QuadRes-2-2 [simp, intro]: QuadRes 2 2
  unfolding QuadRes-def
  unfolding cong-def
  by presburger

```

lemma *Suc-mod-eq[simp]*: $[Suc\ a = Suc\ b] \pmod{2} = [a = b] \pmod{2}$
using *Suc-eq-plus1-left cong-add-lcancel-nat* **by** *presburger*

lemma *div-cancel-aux*: $c\ dvd\ a \implies (d + a * b)\ div\ c = (d\ div\ c) + a\ div\ c * b$
for $a\ b\ c :: nat$
by (*metis div-plus-div-distrib-dvd-right dvd-div-mult dvd-trans dvd-triv-left*)

corollary *div-cancel-Suc*: $c\ dvd\ a \implies 1 < c \implies Suc\ (a * b)\ div\ c = a\ div\ c * b$
using *div-cancel-aux[where d = 1]* **by** *fastforce*

lemma *cong-aux-eq-1*: $odd\ p \implies [(p - 1)\ div\ 2 - p\ div\ 4 = (p^2 - 1)\ div\ 8] \pmod{2}$
for $p :: nat$

proof (*induction p rule: nat-less-induct*)
case $(1\ n)$

consider $n = 1 \mid n > 1$ **using** *odd-pos[OF <odd n>]* **by** *linarith*

then show *?case proof (cases)*
assume $n > 1$

then obtain m **where** $m: m = n - 2$ **and** $m': odd\ m\ m < n$ **using** *<odd n>*
by *simp*

then obtain b **where** $b: m = 2 * b + 1$ **using** *oddE* **by** *blast*

have *IH*: $[(m - 1)\ div\ 2 - m\ div\ 4 = (m^2 - 1)\ div\ 8] \pmod{2}$ **using** *1.IH*
 m' **by** *simp*

have [*simp*]: $n = 2 * b + 1 + 2$ **using** $m\ <n > 1\ b$ **by** *auto*

have $*$: $(n^2 - 1)\ div\ 8 = ((n - 2)^2 - 1)\ div\ 8 + (n - 1)\ div\ 2$
unfolding *power2-sum power2-eq-square* **by** *simp*

have $[(n - 1)\ div\ 2 - n\ div\ 4 = (n - 2 - 1)\ div\ 2 - (n - 2)\ div\ 4 + (n - 1)\ div\ 2] \pmod{2}$

by (*rule cong-sym, cases even b*) (*auto simp: cong-altdef-nat div-cancel-Suc elim: oddE*)

also have $[(n - 2 - 1)\ div\ 2 - (n - 2)\ div\ 4 + (n - 1)\ div\ 2 = (n^2 - 1)\ div\ 8] \pmod{2}$

using *IH cong-add-rcancel-nat unfolding * m* **by** *presburger*

finally show *?thesis* .

qed *simp*
qed

lemma *cong-2-pow[intro]*: $(-1 :: int)^a = (-1)^b$ **if** $[a = b] \pmod{2}$ **for** $a\ b :: nat$

proof -

have *even a = even b*

by (*simp add: cong-dvd-iff that*)

then show *?thesis* **by** *auto*
qed

lemma *card-Int*: $\text{card } (A \cap B) = \text{card } A - \text{card } (A - B)$ **if** *finite A*
by (*metis Diff-Diff-Int Diff-subset card-Diff-subset finite-Diff that*)

Proofs are inspired by [3].

theorem *supplement2-Legendre*:

fixes $p :: \text{int}$
assumes $p > 2$ *prime p*
shows $\text{Legendre } 2 \ p = (-1)^{\wedge}(((\text{nat } p)^{\wedge} 2 - 1) \ \text{div } 8)$
proof –
interpret *GAUSS* $\text{nat } p \ 2$
using *assms*
unfolding *GAUSS-def prime-int-nat-transfer*
by (*simp add: two-cong-0-iff-int*)

have $\text{card } E = \text{card } ((\lambda x. x * 2 \ \text{mod } p) \ ‘ \{0 < .. (p - 1) \ \text{div } 2\} \cap \{(p - 1) \ \text{div } 2 < ..\})$ (**is** $= \text{card } ?A$)
unfolding *E-def C-def B-def A-def image-image* **using** *assms* **by** *simp*
also have $(\lambda x. x * 2 \ \text{mod } p) \ ‘ \{0 < .. (p - 1) \ \text{div } 2\} = ((*) \ 2) \ ‘ \{0 < .. (p - 1) \ \text{div } 2\}$
by (*intro image-cong*) *auto*
also have $\text{card } (\dots \cap \{(p - 1) \ \text{div } 2 < ..\}) = \text{nat } ((p - 1) \ \text{div } 2) - \text{card } ((*) \ 2 \ ‘ \{0 < .. (p - 1) \ \text{div } 2\} - \{(p - 1) \ \text{div } 2 < ..\})$
using *assms* **by** (*subst card-Int*) (*auto simp: card-image inj-onI*)
also have $\text{card } (((*) \ 2) \ ‘ \{0 < .. (p - 1) \ \text{div } 2\} - \{(p - 1) \ \text{div } 2 < ..\}) = \text{card } \{0 < .. (p \ \text{div } 4)\}$
by (*rule sym, intro bij-betw-same-card[of (*) 2] bij-betw-imageI inj-onI*)
(*insert assms prime-odd-int[of p], auto simp: image-def elim!: oddE*)
also have $\dots = \text{nat } p \ \text{div } 4$ **using** *assms* **by** *simp*
also have $\text{nat } ((p - 1) \ \text{div } 2) - \text{nat } p \ \text{div } 4 = \text{nat } ((p - 1) \ \text{div } 2 - p \ \text{div } 4)$
using *assms* **by** (*simp add: nat-diff-distrib nat-div-distrib*)
finally have $\text{card } E = \dots$.

then have $\text{Legendre } 2 \ p = (-1)^{\wedge} \text{nat } ((p - 1) \ \text{div } 2 - p \ \text{div } 4)$
using *gauss-lemma assms* **by** *simp*
also have $\text{nat } ((p - 1) \ \text{div } 2 - p \ \text{div } 4) = (\text{nat } p - 1) \ \text{div } 2 - \text{nat } p \ \text{div } 4$
using *assms* **by** (*simp add: nat-div-distrib nat-diff-distrib*)
also have $(-1)^{\wedge} \dots = ((-1)^{\wedge}(((\text{nat } p)^{\wedge} 2 - 1) \ \text{div } 8)) :: \text{int}$
using *cong-aux-eq-1[of nat p] odd-p* **by** *blast*
finally show *?thesis* .
qed

theorem *supplement1-Legendre*:

prime p $\implies 2 < p \implies \text{Legendre } (-1) \ p = (-1)^{\wedge}((p-1) \ \text{div } 2)$
using *euler-criterion[of p - 1] Legendre-in-cong-eq[symmetric, of p]*
by (*simp add: minus-one-power-iff*)

lemma *QuadRes-1-right* [*intro, simp*]: *QuadRes p 1*
by (*metis QuadRes-def cong-def power-one*)

lemma *Legendre-1-left* [*simp*]: *prime p \implies Legendre 1 p = 1*
by (*auto simp add: Legendre-def cong-iff-dvd-diff not-prime-unit*)

lemma *cong-eq-0-not-coprime*: *prime p \implies [a = 0] (mod p) \implies \neg coprime a p* **for**
a p :: int
unfolding *cong-iff-dvd-diff prime-int-iff*
by *auto*

lemma *not-coprime-cong-eq-0*: *prime p \implies \neg coprime a p \implies [a = 0] (mod p)*
for *a p :: int*
unfolding *cong-iff-dvd-diff*
using *prime-imp-coprime[of p a]*
by (*auto simp: coprime-commute*)

lemma *prime-cong-eq-0-iff*: *prime p \implies [a = 0] (mod p) \longleftrightarrow \neg coprime a p* **for**
a p :: int
using *not-coprime-cong-eq-0[of p a] cong-eq-0-not-coprime[of p a]*
by *auto*

lemma *Legendre-eq-0-iff* [*simp*]: *prime p \implies Legendre a p = 0 \longleftrightarrow \neg coprime a p*
unfolding *Legendre-def* **by** (*auto simp: prime-cong-eq-0-iff*)

lemma *Legendre-prod-mset* [*simp*]: *prime p \implies Legendre (prod-mset M) p =*
($\prod_{q \in \#M} \text{Legendre } q \text{ } p$)
by (*induction M*) *simp-all*

lemma *Legendre-0-eq-0* [*simp*]: *Legendre 0 p = 0* **unfolding** *Legendre-def* **by** *auto*

lemma *Legendre-values*: *Legendre p q \in {1, -1, 0}*
unfolding *Legendre-def* **by** *auto*

end

2 Auxiliary Material

theory *Algebraic-Auxiliaries*
imports
HOL-Algebra.Algebra
HOL-Computational-Algebra.Squarefree
HOL-Number-Theory.Number-Theory
begin

hide-const (**open**) *Divisibility.prime*

lemma *sum-of-bool-eq-card*:

assumes *finite S*
shows $(\sum a \in S. \text{of-bool } (P a)) = \text{real } (\text{card } \{a \in S . P a \})$
proof –
have $(\sum a \in S. \text{of-bool } (P a) :: \text{real}) = (\sum a \in \{x \in S. P x\}. 1)$
using *assms* **by** (*intro sum.mono-neutral-cong-right*) *auto*
thus *?thesis* **by** *simp*
qed

lemma *mod-natE*:
fixes *a n b :: nat*
assumes *a mod n = b*
shows $\exists l. a = n * l + b$
using *assms mod-mult-div-eq[of a n]* **by** (*metis add.commute*)

lemma (*in group*) *r-coset-is-image*: $H \#> a = (\lambda x. x \otimes a) ' H$
unfolding *r-coset-def image-def*
by *blast*

lemma (*in group*) *FactGroup-order*:
assumes *subgroup H G finite H*
shows $\text{order } G = \text{order } (G \text{ Mod } H) * \text{card } H$
using *lagrange assms* **unfolding** *FactGroup-def order-def* **by** *simp*

corollary (*in group*) *FactGroup-order-div*:
assumes *subgroup H G finite H*
shows $\text{order } (G \text{ Mod } H) = \text{order } G \text{ div } \text{card } H$
using *assms FactGroup-order subgroupE(2)[OF <subgroup H G>]* **by** (*auto simp: order-def*)

lemma *group-hom-imp-group-hom-image*:
assumes *group-hom G G h*
shows *group-hom G (G (carrier := h ' carrier G)) h*
using *group-hom.axioms[OF assms] group-hom.img-is-subgroup[OF assms] group.subgroup-imp-group*
by (*auto intro!: group-hom.intro simp: group-hom.axioms-def hom-def*)

theorem *homomorphism-thm*:
assumes *group-hom G G h*
shows $G \text{ Mod } \text{kernel } G (G (carrier := h ' carrier G)) h \cong G (carrier := h ' carrier G)$
by (*intro group-hom.FactGroup-iso group-hom-imp-group-hom-image assms*) *simp*

lemma *is-iso-imp-same-card*:
assumes $H \cong G$
shows $\text{order } H = \text{order } G$
proof –
from *assms* **obtain** *h* **where** *bij-betw h (carrier H) (carrier G)*
unfolding *is-iso-def iso-def*
by *blast*

then show *?thesis*
unfolding *order-def*
by (*rule bij-betw-same-card*)
qed

corollary *homomorphism-thm-order:*

assumes *group-hom G G h*
shows $order (G(\backslash carrier := h \text{ ' } carrier G)) * card (kernel G (G(\backslash carrier := h \text{ ' } carrier G)) h) = order G$

proof –

have $order (G(\backslash carrier := h \text{ ' } carrier G)) = order (G Mod (kernel G (G(\backslash carrier := h \text{ ' } carrier G)) h))$

using *is-iso-imp-same-card[OF homomorphism-thm] <group-hom G G h>*
by *fastforce*

moreover have *group G using <group-hom G G h> group-hom.axioms by blast*

ultimately show *?thesis*

using *<group-hom G G h> and group-hom-imp-group-hom-image[OF <group-hom G G h>]*

unfolding *FactGroup-def*

by (*simp add: group.lagrange group-hom.subgroup-kernel order-def*)

qed

lemma (*in group-hom*) *kernel-subset: kernel G H h \subseteq carrier G*

using *subgroup-kernel G.subgroupE(1) by blast*

lemma (*in group*) *proper-subgroup-imp-bound-on-card:*

assumes $H \subset carrier G$ *subgroup H G finite (carrier G)*

shows $card H \leq order G \div 2$

proof –

from *<finite (carrier G)> have finite (rcosets H)*

by (*simp add: RCOSETS-def*)

note *subgroup.subgroup-in-rcosets[OF <subgroup H G> is-group]*

then obtain *J where $J \neq H$ $J \in rcosets H$*

using *rcosets-part-G[OF <subgroup H G>] and <H \subset carrier G>*

by (*metis Sup-le-iff inf.absorb-iff2 inf.idem inf.strict-order-iff*)

then have $2 \leq card (rcosets H)$

using *<H \in rcosets H> card-mono[OF <finite (rcosets H)>, of {H, J}]*

by *simp*

then show *?thesis*

using *mult-le-mono[of 2 card (rcosets H) card H card H]*

unfolding *lagrange[OF <subgroup H G>]*

by *force*

qed

lemma *cong-exp-trans*[*trans*]:

$$\begin{aligned} [a \wedge b = c] \text{ (mod } n) &\Longrightarrow [a = d] \text{ (mod } n) \Longrightarrow [d \wedge b = c] \text{ (mod } n) \\ [c = a \wedge b] \text{ (mod } n) &\Longrightarrow [a = d] \text{ (mod } n) \Longrightarrow [c = d \wedge b] \text{ (mod } n) \end{aligned}$$

using *cong-pow cong-sym cong-trans* by *blast+*

lemma *cong-exp-mod*[*simp*]:

$$\begin{aligned} [(a \text{ mod } n) \wedge b = c] \text{ (mod } n) &\longleftrightarrow [a \wedge b = c] \text{ (mod } n) \\ [c = (a \text{ mod } n) \wedge b] \text{ (mod } n) &\longleftrightarrow [c = a \wedge b] \text{ (mod } n) \end{aligned}$$

by (*auto simp add: cong-def mod-simps*)

lemma *cong-mult-mod*[*simp*]:

$$\begin{aligned} [(a \text{ mod } n) * b = c] \text{ (mod } n) &\longleftrightarrow [a * b = c] \text{ (mod } n) \\ [a * (b \text{ mod } n) = c] \text{ (mod } n) &\longleftrightarrow [a * b = c] \text{ (mod } n) \end{aligned}$$

by (*auto simp add: cong-def mod-simps*)

lemma *cong-add-mod*[*simp*]:

$$\begin{aligned} [(a \text{ mod } n) + b = c] \text{ (mod } n) &\longleftrightarrow [a + b = c] \text{ (mod } n) \\ [a + (b \text{ mod } n) = c] \text{ (mod } n) &\longleftrightarrow [a + b = c] \text{ (mod } n) \\ [\sum_{i \in A} i \text{ mod } n = c] \text{ (mod } n) &\longleftrightarrow [\sum_{i \in A} i = c] \text{ (mod } n) \end{aligned}$$

by (*auto simp add: cong-def mod-simps*)

lemma *cong-add-trans*[*trans*]:

$$\begin{aligned} [a = b + x] \text{ (mod } n) &\Longrightarrow [x = y] \text{ (mod } n) \Longrightarrow [a = b + y] \text{ (mod } n) \\ [a = x + b] \text{ (mod } n) &\Longrightarrow [x = y] \text{ (mod } n) \Longrightarrow [a = y + b] \text{ (mod } n) \\ [b + x = a] \text{ (mod } n) &\Longrightarrow [x = y] \text{ (mod } n) \Longrightarrow [b + y = a] \text{ (mod } n) \\ [x + b = a] \text{ (mod } n) &\Longrightarrow [x = y] \text{ (mod } n) \Longrightarrow [y + b = a] \text{ (mod } n) \end{aligned}$$

unfolding *cong-def*
using *mod-simps*(1, 2)
by *metis+*

lemma *cong-mult-trans*[*trans*]:

$$\begin{aligned} [a = b * x] \text{ (mod } n) &\Longrightarrow [x = y] \text{ (mod } n) \Longrightarrow [a = b * y] \text{ (mod } n) \\ [a = x * b] \text{ (mod } n) &\Longrightarrow [x = y] \text{ (mod } n) \Longrightarrow [a = y * b] \text{ (mod } n) \\ [b * x = a] \text{ (mod } n) &\Longrightarrow [x = y] \text{ (mod } n) \Longrightarrow [b * y = a] \text{ (mod } n) \\ [x * b = a] \text{ (mod } n) &\Longrightarrow [x = y] \text{ (mod } n) \Longrightarrow [y * b = a] \text{ (mod } n) \end{aligned}$$

unfolding *cong-def*
using *mod-simps*(4, 5)
by *metis+*

lemma *cong-diff-trans*[*trans*]:

$$\begin{aligned} [a = b - x] \text{ (mod } n) &\Longrightarrow [x = y] \text{ (mod } n) \Longrightarrow [a = b - y] \text{ (mod } n) \\ [a = x - b] \text{ (mod } n) &\Longrightarrow [x = y] \text{ (mod } n) \Longrightarrow [a = y - b] \text{ (mod } n) \\ [b - x = a] \text{ (mod } n) &\Longrightarrow [x = y] \text{ (mod } n) \Longrightarrow [b - y = a] \text{ (mod } n) \\ [x - b = a] \text{ (mod } n) &\Longrightarrow [x = y] \text{ (mod } n) \Longrightarrow [y - b = a] \text{ (mod } n) \end{aligned}$$

for $a :: 'a :: \{\text{unique-euclidean-semiring, euclidean-ring-cancel}\}$
unfolding *cong-def*
by (*metis mod-diff-eq*)+

lemma *eq-imp-eq-mod-int*: $a = b \Longrightarrow [a = b] \text{ (mod } m)$ for $a b :: \text{int}$ by *simp*

lemma *eq-imp-eq-mod-nat*: $a = b \implies [a = b] \text{ (mod } m)$ **for** $a\ b :: \text{nat}$ **by** *simp*

lemma *cong-pow-I*: $a = b \implies [x^a = x^b] \text{ (mod } n)$ **by** *simp*

lemma *gre1I*: $(n = 0 \implies \text{False}) \implies (1 :: \text{nat}) \leq n$
by *presburger*

lemma *gre1I-nat*: $(n = 0 \implies \text{False}) \implies (\text{Suc } 0 :: \text{nat}) \leq n$
by *presburger*

lemma *totient-less-not-prime*:
assumes $\neg \text{prime } n\ 1 < n$
shows $\text{totient } n < n - 1$
using *totient-imp-prime totient-less assms*
by (*metis One-nat-def Suc-pred le-less-trans less-SucE zero-le-one*)

lemma *power2-diff-nat*: $x \geq y \implies (x - y)^2 = x^2 + y^2 - 2 * x * y$ **for** $x\ y :: \text{nat}$
by (*simp add: algebra-simps power2-eq-square mult-2-right*)
(*meson Nat.diff-diff-right le-add2 le-trans mult-le-mono order-refl*)

lemma *square-inequality*: $1 < n \implies (n + n) \leq (n * n)$ **for** $n :: \text{nat}$
by (*metis Suc-eq-plus1-left Suc-leI mult-le-mono1 semiring-normalization-rules(4)*)

lemma *square-one-cong-one*:
assumes $[x = 1] \text{ (mod } n)$
shows $[x^2 = 1] \text{ (mod } n)$
using *assms cong-pow* **by** *fastforce*

lemma *cong-square-alt-int*:
 $\text{prime } p \implies [a * a = 1] \text{ (mod } p) \implies [a = 1] \text{ (mod } p) \vee [a = p - 1] \text{ (mod } p)$
for $a\ p :: 'a :: \{\text{normalization-semidom, linordered-idom, unique-euclidean-ring}\}$
using *dvd-add-triv-right-iff[of p a - (p - 1)]*
by (*auto simp add: cong-iff-dvd-diff square-diff-one-factored dest!: prime-dvd-multD*)

lemma *cong-square-alt*:
 $\text{prime } p \implies [a * a = 1] \text{ (mod } p) \implies [a = 1] \text{ (mod } p) \vee [a = p - 1] \text{ (mod } p)$
for $a\ p :: \text{nat}$
using *cong-square-alt-int[of int p int a] prime-nat-int-transfer[of p] prime-gt-1-nat[of p]*
by (*simp flip: cong-int-iff add: of-nat-diff*)

lemma *square-minus-one-cong-one*:
fixes $n\ x :: \text{nat}$
assumes $1 < n\ [x = n - 1] \text{ (mod } n)$
shows $[x^2 = 1] \text{ (mod } n)$
proof –
have $[x^2 = (n - 1) * (n - 1)] \text{ (mod } n)$
using *cong-mult[OF assms(2) assms(2)]*
by (*simp add: algebra-simps power2-eq-square*)

also have $[(n - 1) * (n - 1) = \text{Suc } (n * n) - (n + n)] \text{ (mod } n)$
using *power2-diff-nat*[*of 1 n*] $\langle 1 < n \rangle$
by (*simp add: algebra-simps power2-eq-square*)

also have $[\text{Suc } (n * n) - (n + n) = \text{Suc } (n * n)] \text{ (mod } n)$
proof –

have $n * n + 0 * n = n * n$ **by** *linarith*
moreover have $n * n - (n + n) + (n + n) = n * n$
using *square-inequality*[*OF*] $\langle 1 < n \rangle$ *le-add-diff-inverse2* **by** *blast*
moreover have $(\text{Suc } 0 + 1) * n = n + n$
by *simp*
ultimately show *?thesis*
using *square-inequality*[*OF*] $\langle 1 < n \rangle$
by (*metis (no-types) Suc-diff-le add-Suc cong-iff-lin-nat*)

qed

also have $[\text{Suc } (n * n) = 1] \text{ (mod } n)$
using *cong-to-1'-nat* **by** *auto*

finally show *?thesis* .
qed

lemma *odd-prime-gt-2-int*:
 $2 < p$ **if** *odd p prime p* **for** $p :: \text{int}$
using *prime-ge-2-int*[*OF*] $\langle \text{prime } p \rangle$ $\langle \text{odd } p \rangle$
by (*cases p = 2*) *auto*

lemma *odd-prime-gt-2-nat*:
 $2 < p$ **if** *odd p prime p* **for** $p :: \text{nat}$
using *prime-ge-2-nat*[*OF*] $\langle \text{prime } p \rangle$ $\langle \text{odd } p \rangle$
by (*cases p = 2*) *auto*

lemma *gt-one-imp-gt-one-power-if-coprime*:
 $1 \leq x \implies 1 < n \implies \text{coprime } x \ n \implies 1 \leq x \wedge (n - 1) \text{ mod } n$
by (*rule gre1I*) (*auto simp: coprime-commute dest: coprime-absorb-left*)

lemma *residue-one-dvd*: $a \text{ mod } n = 1 \implies n \text{ dvd } a - 1$ **for** $a \ n :: \text{nat}$
by (*fastforce intro!: cong-to-1-nat simp: cong-def*)

lemma *coprimeI-power-mod*:
fixes $x \ r \ n :: \text{nat}$
assumes $x \wedge r \text{ mod } n = 1 \ r \neq 0 \ n \neq 0$
shows *coprime x n*
proof –
have *coprime (x ^ r mod n) n*
using *coprime-1-right* $\langle x \wedge r \text{ mod } n = 1 \rangle$
by (*simp add: coprime-commute*)
thus *?thesis* **using** $\langle r \neq 0 \rangle \langle n \neq 0 \rangle$ **by** *simp*

qed

lemma *prime-dvd-choose*:

assumes $0 < k < p$ *prime* p

shows $p \text{ dvd } (p \text{ choose } k)$

proof –

have $k \leq p$ **using** $\langle k < p \rangle$ **by** *auto*

have $p \text{ dvd fact } p$ **using** $\langle \text{prime } p \rangle$ **by** (*simp add: prime-dvd-fact-iff*)

moreover have $\neg p \text{ dvd fact } k * \text{fact } (p - k)$

unfolding *prime-dvd-mult-iff*[*OF* $\langle \text{prime } p \rangle$] *prime-dvd-fact-iff*[*OF* $\langle \text{prime } p \rangle$]

using *assms* **by** *simp*

ultimately show *?thesis*

unfolding *binomial-fact-lemma*[*OF* $\langle k \leq p \rangle$, *symmetric*]

using *assms* *prime-dvd-multD* **by** *blast*

qed

lemma *cong-eq-0-I*: $(\forall i \in A. [f \ i \ \text{mod } n = 0] \ (\text{mod } n)) \implies [\sum i \in A. f \ i = 0] \ (\text{mod } n)$

using *cong-sum* **by** *fastforce*

lemma *power-mult-cong*:

assumes $[x^n = a] \ (\text{mod } m)$ $[y^n = b] \ (\text{mod } m)$

shows $[(x*y)^n = a*b] \ (\text{mod } m)$

using *assms* *cong-mult*[*of* $x^n \ a \ m \ y^n \ b$] *power-mult-distrib*

by *metis*

lemma

fixes $n :: \text{nat}$

assumes $n > 1$

shows *odd-pow-cong*: $\text{odd } m \implies [(n - 1)^m = n - 1] \ (\text{mod } n)$

and *even-pow-cong*: $\text{even } m \implies [(n - 1)^m = 1] \ (\text{mod } n)$

proof (*induction m*)

case (*Suc m*)

case 1

with *Suc* **have** *IH*: $[(n - 1)^m = 1] \ (\text{mod } n)$ **by** *auto*

show *?case* **using** $\langle 1 < n \rangle$ *cong-mult*[*OF* *cong-refl IH*] **by** *simp*

next

case (*Suc m*)

case 2

with *Suc* **have** *IH*: $[(n - 1)^m = n - 1] \ (\text{mod } n)$ **by** *auto*

show *?case*

using *cong-mult*[*OF* *cong-refl IH*, *of* $(n - 1)$] **and** *square-minus-one-cong-one*[*OF* $\langle 1 < n \rangle$, *of* $n - 1$]

by (*auto simp: power2-eq-square intro: cong-trans*)

qed *simp-all*

lemma *cong-mult-uneq'*:

fixes $a :: 'a::\{unique-euclidean-ring, ring-gcd\}$
assumes *coprime d a*
shows $[b \neq c] \pmod{a} \implies [d = e] \pmod{a} \implies [b * d \neq c * e] \pmod{a}$
using *cong-mult-rcancel[OF assms]*
using *cong-trans[of b*d c*e a c*d]*
using *cong-scalar-left cong-sym* by *blast*

lemma *p-coprime-right-nat*: *prime p* \implies *coprime a p* = $(\neg p \text{ dvd } a)$ for $p a :: \text{nat}$
by (*meson coprime-absorb-left coprime-commute not-prime-unit prime-imp-coprime-nat*)

lemma *squarefree-mult-imp-coprime*:

assumes *squarefree (a * b :: 'a :: semiring-gcd)*
shows *coprime a b*
proof (*rule coprimeI*)
fix l assume $l \text{ dvd } a \ l \text{ dvd } b$
then obtain $a' b'$ where $a = l * a' \ b = l * b'$
by (*auto elim!: dvdE*)
with *assms* have *squarefree (l² * (a' * b'))*
by (*simp add: power2-eq-square mult-ac*)
thus $l \text{ dvd } 1$ by (*rule squarefreeD*) *auto*
qed

lemma *prime-divisor-exists-strong*:

fixes $m :: \text{int}$
assumes $m > 1 \ \neg\text{prime } m$
shows $\exists n k. m = n * k \wedge 1 < n \wedge n < m \wedge 1 < k \wedge k < m$
proof –
from *assms* obtain $n k$ where *nk*: $n * k > 1 \ n \geq 0 \ m = n * k \ n \neq 1 \ n \neq 0 \ k \neq 1$
using *assms* *unfolding prime-int-iff dvd-def* by *auto*
from *nk* have $n > 1$ by *linarith*

from *nk* *assms* have $n * k > 0$ by *simp*
with $\langle n \geq 0 \rangle$ have $k > 0$
using *zero-less-mult-pos* by *force*
with $\langle k \neq 1 \rangle$ have $k > 1$ by *linarith*
from *nk* have $n > 1$ by *linarith*

from $\langle k > 1 \rangle$ *nk* have $n < m \ k < m$ by *simp-all*
with *nk* $\langle k > 1 \rangle \langle n > 1 \rangle$ show *?thesis* by *blast*
qed

lemma *prime-divisor-exists-strong-nat*:

fixes $m :: \text{nat}$
assumes $1 < m \ \neg\text{prime } m$

shows $\exists p k. m = p * k \wedge 1 < p \wedge p < m \wedge 1 < k \wedge k < m \wedge \text{prime } p$
proof –
obtain p **where** $p\text{-def: prime } p \ p \ \text{dvd } m \ p \neq m \ 1 < p$
using $\text{assms prime-prime-factor and prime-gt-1-nat}$
by blast

moreover define k **where** $k = m \ \text{div } p$
with $\langle p \ \text{dvd } m \rangle$ **have** $m = p * k$ **by** simp

moreover have $p < m$
using $\langle p \neq m \rangle \ \text{dvd-imp-le}[OF \ \langle p \ \text{dvd } m \rangle]$ **and** $\langle m > 1 \rangle$
by simp

moreover have $1 < k \ k < m$
using $\langle 1 < m \rangle \ \langle 1 < p \rangle$ **and** $\langle p \neq m \rangle$
unfolding $\langle m = p * k \rangle$
by $(\text{force intro: Suc-lessI Nat.gr0I})+$

ultimately show $?thesis$ **using** $\langle 1 < m \rangle$ **by** blast
qed

lemma $\text{prime-factorization-eqI}$:
assumes $\bigwedge p. p \in \# P \implies \text{prime } p \ \text{prod-mset } P = n$
shows $\text{prime-factorization } n = P$
using $\text{prime-factorization-prod-mset-primes}[of P]$ assms **by** simp

lemma $\text{prime-factorization-prime-elem}$:
assumes $\text{prime-elem } p$
shows $\text{prime-factorization } p = \{\# \text{normalize } p \#\}$
proof –
have $\text{prime-factorization } p = \text{prime-factorization } (\text{normalize } p)$
by $(\text{metis normalize-idem prime-factorization-cong})$
also have $\dots = \{\# \text{normalize } p \#\}$
by $(\text{rule prime-factorization-prime})$ $(\text{use assms in auto})$
finally show $?thesis$.
qed

lemma $\text{size-prime-factorization-eq-Suc-0-iff}$ $[simp]$:
fixes $n :: 'a :: \text{factorial-semiring-multiplicative}$
shows $\text{size } (\text{prime-factorization } n) = \text{Suc } 0 \iff \text{prime-elem } n$
proof
assume $\text{size: size } (\text{prime-factorization } n) = \text{Suc } 0$
hence $[simp]: n \neq 0$ **by** auto
from size **obtain** p **where** $*$: $\text{prime-factorization } n = \{\# p \#\}$
by $(\text{auto elim!: size-mset-SucE})$
hence $p: p \in \text{prime-factors } n$ **by** auto

have $\text{prime-elem } (\text{normalize } p)$

using p **by** (*auto simp: in-prime-factors-iff*)
also have $p = \text{prod-mset (prime-factorization } n)$
using $*$ **by** *simp*
also have $\text{normalize } \dots = \text{normalize } n$
by (*rule prod-mset-prime-factorization-weak*) *auto*
finally show $\text{prime-elem } n$ **by** *simp*
qed (*auto simp: prime-factorization-prime-elem*)

lemma *squarefree-prime-elem* [*simp, intro*]:
fixes $p :: 'a :: \text{algebraic-semidom}$
assumes $\text{prime-elem } p$
shows $\text{squarefree } p$
proof (*rule squarefreeI*)
fix x **assume** $x^2 \text{ dvd } p$
show $\text{is-unit } x$
proof (*rule ccontr*)
assume $\neg \text{is-unit } x$
hence $\neg \text{is-unit } (x^2)$
by (*simp add: is-unit-power-iff*)
from *assms and this and* $\langle x^2 \text{ dvd } p \rangle$ **have** $\text{prime-elem } (x^2)$
by (*rule prime-elem-mono*)
thus *False* **by** (*simp add: prime-elem-power-iff*)
qed
qed

lemma *squarefree-prime* [*simp, intro*]: $\text{prime } p \implies \text{squarefree } p$
by *auto*

lemma *not-squarefree-primpow*:
assumes $\text{primpow } n$
shows $\text{squarefree } n \longleftrightarrow \text{prime } n$
using *assms* **by** (*auto simp: primpow-def squarefree-power-iff prime-power-iff*)

lemma *prime-factorization-normalize* [*simp*]:
 $\text{prime-factorization (normalize } n) = \text{prime-factorization } n$
by (*rule prime-factorization-cong*) *auto*

lemma *one-prime-factor-iff-primpow*:
fixes $n :: 'a :: \text{factorial-semiring-multiplicative}$
shows $\text{card (prime-factors } n) = \text{Suc } 0 \longleftrightarrow \text{primpow (normalize } n)$
proof
assume $\text{primpow (normalize } n)$
then obtain $p \ k$ **where** $\text{pk: prime } p \ \text{normalize } n = p \wedge k \ k > 0$
by (*auto simp: primpow-def*)
hence $\text{card (prime-factors (normalize } n)) = \text{Suc } 0$
by (*subst pk*) (*simp add: prime-factors-power prime-factorization-prime*)
thus $\text{card (prime-factors } n) = \text{Suc } 0$


```

    by simp
next
assume *: card (prime-factors n) = Suc 0
from * have (∏ p∈prime-factors n. p ^ multiplicity p n) = normalize n
  by (intro prod-prime-factors) auto
also from * have card (prime-factors n) = 1 by simp
then obtain p where p: prime-factors n = {p}
  by (elim card-1-singletonE)
finally have normalize n = p ^ multiplicity p n
  by simp
moreover from p have prime p multiplicity p n > 0
  by (auto simp: prime-factors-multiplicity)
ultimately show primepow (normalize n)
  unfolding primepow-def by blast
qed

```

```

lemma squarefree-imp-prod-prime-factors-eq:
  fixes x :: 'a :: factorial-semiring-multiplicative
  assumes squarefree x
  shows ∏ (prime-factors x) = normalize x
proof -
  from assms have [simp]: x ≠ 0 by auto
  have (∏ p∈prime-factors x. p ^ multiplicity p x) = normalize x
    by (intro prod-prime-factors) auto
  also have (∏ p∈prime-factors x. p ^ multiplicity p x) = (∏ p∈prime-factors x.
p)
  using assms by (intro prod.cong refl) (auto simp: squarefree-factorial-semiring')
  finally show ?thesis by simp
qed

```

end

3 The Jacobi Symbol

```

theory Jacobi-Symbol
imports
  Legendre-Symbol
  Algebraic-Auxiliaries
begin

```

The Jacobi symbol is a generalisation of the Legendre symbol to non-primes [4, 2]. It is defined as

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{k_1} \cdots \left(\frac{a}{p_l}\right)^{k_l}$$

where $\left(\frac{a}{p}\right)$ denotes the Legendre symbol, a is an integer, n is an odd natural number and $p_1^{k_1} \cdots p_l^{k_l}$ is its prime factorisation.

There is, however, a fairly natural generalisation to all non-zero integers for n . It is less clear what a good choice for $n = 0$ is; Mathematica and Maxima adopt the convention that $\left(\frac{\pm 1}{0}\right) = 1$ and $\left(\frac{a}{0}\right) = 0$ otherwise. However, we chose the slightly different convention $\left(\frac{a}{0}\right) = 0$ for *all* a because then the Jacobi symbol is completely multiplicative in both arguments without any restrictions.

definition *Jacobi* :: $int \Rightarrow int \Rightarrow int$ **where**
Jacobi $a\ n = (if\ n = 0\ then\ 0\ else$
 $(\prod_{p \in \#prime-factorization\ n} Legendre\ a\ p))$

lemma *Jacobi-0-right* [*simp*]: *Jacobi* $a\ 0 = 0$
by (*simp add: Jacobi-def*)

lemma *Jacobi-mult-left* [*simp*]: *Jacobi* $(a * b)\ n = Jacobi\ a\ n * Jacobi\ b\ n$
proof (*cases n = 0*)

case *False*
have *: $\{\# Legendre\ (a * b)\ p \ .\ p \in \# prime-factorization\ n\ \# \} =$
 $\{\# Legendre\ a\ p * Legendre\ b\ p \ .\ p \in \# prime-factorization\ n\ \# \}$
by (*meson Legendre-mult in-prime-factors-imp-prime image-mset-cong*)

show *?thesis* **using** *False unfolding Jacobi-def * prod-mset.distrib* **by** *auto*
qed *auto*

lemma *Jacobi-mult-right* [*simp*]: *Jacobi* $a\ (n * m) = Jacobi\ a\ n * Jacobi\ a\ m$
by (*cases m = 0; cases n = 0*)
(auto simp: Jacobi-def prime-factorization-mult)

lemma *prime-p-Jacobi-eq-Legendre*[*intro!*]: *prime* $p \implies Jacobi\ a\ p = Legendre\ a\ p$
unfolding *Jacobi-def prime-factorization-prime* **by** *simp*

lemma *Jacobi-mod* [*simp*]: *Jacobi* $(a\ mod\ m)\ n = Jacobi\ a\ n$ **if** $n\ dvd\ m$
proof –

have *: $\{\# Legendre\ (a\ mod\ m)\ p \ .\ p \in \# prime-factorization\ n\ \# \} =$
 $\{\# Legendre\ a\ p \ .\ p \in \# prime-factorization\ n\ \# \}$ **using** *that*
by (*intro image-mset-cong, subst Legendre-mod*)
(auto intro: dvd-trans[OF in-prime-factors-imp-dvd])
thus *?thesis* **by** (*simp add: Jacobi-def*)

qed

lemma *Jacobi-mod-cong*: $[a = b]\ (mod\ n) \implies Jacobi\ a\ n = Jacobi\ b\ n$
by (*metis Jacobi-mod cong-def dvd-refl*)

lemma *Jacobi-1-eq-1* [*simp*]: $p \neq 0 \implies Jacobi\ 1\ p = 1$
by (*simp add: Jacobi-def in-prime-factors-imp-prime cong: image-mset-cong*)

lemma *Jacobi-eq-0-not-coprime*:
assumes $n \neq 0 \neg coprime\ a\ n$
shows $Jacobi\ a\ n = 0$

proof –

from *assms* **have** $\exists p. p \text{ dvd } \text{gcd } a \ n \wedge \text{prime } p$
by (*intro prime-divisor-exists*) *auto*
then obtain p **where** $p: p \text{ dvd } a \ p \text{ dvd } n \ \text{prime } p$
by *auto*
hence *Legendre* $a \ p = 0$ **using** *assms*
by (*auto simp: prime-int-iff*)
thus *?thesis* **using** p *assms*
unfolding *Jacobi-def*
by (*auto simp: image-iff prime-factors-dvd*)

qed

lemma *Jacobi-p-eq-2*[*simp*]: $n > 0 \implies \text{Jacobi } a \ (2^n) = a \bmod 2$
by (*auto simp add: Jacobi-def prime-factorization-prime-power*)

lemma *Jacobi-prod-mset*[*simp*]: $n \neq 0 \implies \text{Jacobi } (\text{prod-mset } M) \ n = (\prod_{q \in \#M. \text{Jacobi } q \ n})$
by (*induction M*) *simp-all*

lemma *non-trivial-coprime-neg*:

$1 < a \implies 1 < b \implies \text{coprime } a \ b \implies a \neq b$ **for** $a \ b :: \text{int}$ **by** *auto*

lemma *odd-odd-even*:

fixes $a \ b :: \text{int}$
assumes *odd a odd b*
shows $\text{even } ((a*b-1) \text{ div } 2) = \text{even } ((a-1) \text{ div } 2 + (b-1) \text{ div } 2)$
using *assms* **by** (*auto elim!: oddE simp: algebra-simps*)

lemma *prime-nonprime-wlog* [*case-names primes nonprime sym*]:

assumes $\bigwedge p \ q. \text{prime } p \implies \text{prime } q \implies P \ p \ q$
assumes $\bigwedge p \ q. \neg \text{prime } p \implies P \ p \ q$
assumes $\bigwedge p \ q. P \ p \ q \implies P \ q \ p$
shows $P \ p \ q$
by (*cases prime p; cases prime q*) (*auto intro: assms*)

lemma *Quadratic-Reciprocity-Jacobi*:

fixes $p \ q :: \text{int}$
assumes *coprime p q*
and $2 < p \ 2 < q$
and *odd p odd q*
shows $\text{Jacobi } p \ q * \text{Jacobi } q \ p = (-1)^{\text{nat } ((p-1) \text{ div } 2 * ((q-1) \text{ div } 2))}$
using *assms*

proof (*induction nat p nat q arbitrary: p q*

rule: measure-induct-rule[**where** $f = \lambda(a, b). a + b$, *split-format(complete)*, *simplified*])

case ($1 \ p \ q$)
thus *?case*

```

proof (induction p q rule: prime-nonprime-wlog)
  case (sym p q)
  thus ?case by (simp only: add-ac coprime-commute mult-ac) blast
next
  case (primes p q)
  from ⟨prime p⟩ ⟨prime q⟩ have prime (nat p) prime (nat q) p ≠ q
  using prime-int-nat-transfer primes(4) non-trivial-coprime-neq prime-gt-1-int
  by blast+

  with Quadratic-Reciprocity-int and prime-p-Jacobi-eq-Legendre
  show ?case
  using ⟨prime p⟩ ⟨prime q⟩ primes(5-)
  by presburger
next
  case (nonprime p q)
  from ⟨¬prime p⟩ obtain a b where *: p = a * b 1 < b 1 < a
  using ⟨2 < p⟩ prime-divisor-exists-strong[of p] by auto

  hence odd-ab: odd a odd b using ⟨odd p⟩ by simp-all

  moreover have 2 < b and 2 < a
  using odd-ab and * by presburger+

  moreover have coprime a q and coprime b q using ⟨coprime p q⟩
  unfolding * by simp-all

  ultimately have IH: Jacobi a q * Jacobi q a = (- 1) ^ nat ((a - 1) div 2 *
  ((q - 1) div 2))
  Jacobi b q * Jacobi q b = (- 1) ^ nat ((b - 1) div 2 * ((q -
  1) div 2))
  by (auto simp: * nonprime)

  have pos: 0 < q 0 < p 0 < a 0 < b
  using * ⟨2 < q⟩ by simp-all

  have Jacobi p q * Jacobi q p = (Jacobi a q * Jacobi q a) * (Jacobi b q * Jacobi
  q b)
  using * by simp

  also have ... = (- 1) ^ nat ((a - 1) div 2 * ((q - 1) div 2)) *
  (- 1) ^ nat ((b - 1) div 2 * ((q - 1) div 2))
  using IH by presburger

  also from odd-odd-even[OF odd-ab]
  have ... = (- 1) ^ nat ((p - 1) div 2 * ((q - 1) div 2))
  unfolding * minus-one-power-iff using ⟨2 < q⟩ *
  by (auto simp add: even-nat-iff pos-imp-zdiv-nonneg-iff)

  finally show ?case .

```

qed
qed

lemma *Jacobi-values*: $Jacobi\ p\ q \in \{1, -1, 0\}$

proof (*cases* $q = 0$)

case *False*

hence $|Legendre\ p\ x| = 1$ **if** $x \in \#$ *prime-factorization* q *Jacobi* $p\ q \neq 0$ **for** x

using *that prod-mset-zero-iff Legendre-values*[of $p\ x$]

unfolding *Jacobi-def is-unit-prod-mset-iff set-image-mset*

by *fastforce*

then have *is-unit (prod-mset (image-mset (Legendre p) (prime-factorization q)))*

if *Jacobi* $p\ q \neq 0$

using *that False*

unfolding *Jacobi-def is-unit-prod-mset-iff*

by *auto*

thus *?thesis* **by** (*auto simp: Jacobi-def*)

qed *auto*

lemma *Quadratic-Reciprocity-Jacobi'*:

fixes $p\ q :: int$

assumes *coprime* $p\ q$

and $2 < p\ 2 < q$

and *odd* $p\ odd\ q$

shows $Jacobi\ q\ p = (if\ p\ mod\ 4 = 3 \wedge q\ mod\ 4 = 3\ then\ -1\ else\ 1) * Jacobi$

$p\ q$

proof –

have *aux*: $a \in \{1, -1, 0\} \implies c \neq 0 \implies a*b = c \implies b = c * a$ **for** $b\ c\ a :: int$

by *auto*

from *Quadratic-Reciprocity-Jacobi*[*OF assms*]

have $Jacobi\ q\ p = (-1)^{\wedge nat ((p - 1) div 2 * ((q - 1) div 2))} * Jacobi\ p\ q$

using *Jacobi-values* **by** (*fastforce intro!: aux*)

also have $(-1 :: int)^{\wedge nat ((p - 1) div 2 * ((q - 1) div 2))} = (if\ even\ ((p - 1) div 2) \vee even\ ((q - 1) div 2)\ then\ 1\ else\ -1)$

unfolding *minus-one-power-iff* **using** $\langle 2 < p \rangle\ \langle 2 < q \rangle$

by (*auto simp: even-nat-iff*)

also have $... = (if\ p\ mod\ 4 = 3 \wedge q\ mod\ 4 = 3\ then\ -1\ else\ 1)$

using $\langle odd\ p \rangle\ \langle odd\ q \rangle$ **by** *presburger*

finally show *?thesis* .

qed

lemma *dvd-odd-square*: $8\ dvd\ a^2 - 1$ **if** *odd* a **for** $a :: int$

proof –

```

obtain  $x$  where  $a = 2*x + 1$  using  $\langle \text{odd } a \rangle$  by  $(\text{auto elim: oddE})$ 
thus  $?thesis$ 
  by  $(\text{cases odd } x)$ 
     $(\text{auto elim: oddE simp: power2-eq-square algebra-simps})$ 
qed

lemma  $\text{odd-odd-even'}$ :
  fixes  $a b :: \text{int}$ 
  assumes  $\text{odd } a \text{ odd } b$ 
  shows  $\text{even } (((a * b)^2 - 1) \text{ div } 8) \longleftrightarrow \text{even } (((a^2 - 1) \text{ div } 8) + ((b^2 - 1) \text{ div } 8))$ 
proof -
  obtain  $x$  where  $[simp]: a = 2*x + 1$  using  $\langle \text{odd } a \rangle$  by  $(\text{auto elim: oddE})$ 
  obtain  $y$  where  $[simp]: b = 2*y + 1$  using  $\langle \text{odd } b \rangle$  by  $(\text{auto elim: oddE})$ 
  show  $?thesis$ 
    by  $(\text{cases even } x; \text{cases even } y; \text{elim oddE evenE})$ 
       $(\text{auto simp: power2-eq-square algebra-simps})$ 
qed

lemma  $\text{odd-odd-even-nat'}$ :
  fixes  $a b :: \text{nat}$ 
  assumes  $\text{odd } a \text{ odd } b$ 
  shows  $\text{even } (((a * b)^2 - 1) \text{ div } 8) \longleftrightarrow \text{even } (((a^2 - 1) \text{ div } 8) + ((b^2 - 1) \text{ div } 8))$ 
proof -
  obtain  $x$  where  $[simp]: a = 2*x + 1$  using  $\langle \text{odd } a \rangle$  by  $(\text{auto elim: oddE})$ 
  obtain  $y$  where  $[simp]: b = 2*y + 1$  using  $\langle \text{odd } b \rangle$  by  $(\text{auto elim: oddE})$ 
  show  $?thesis$ 
    by  $(\text{cases even } x; \text{cases even } y; \text{elim oddE evenE})$ 
       $(\text{auto simp: power2-eq-square algebra-simps})$ 
qed

lemma  $\text{supplement2-Jacobi}$ :  $\text{odd } p \implies p > 1 \implies \text{Jacobi } 2 p = (-1)^{\wedge}(((\text{nat } p)^2 - 1) \text{ div } 8)$ 
proof  $(\text{induction } p \text{ rule: prime-divisors-induct})$ 
  case  $(\text{factor } p x)$ 

  then have  $\text{odd } x$  by force

  have  $2 < p$ 
    using  $\langle \text{odd } (p * x) \rangle \text{prime-gt-1-int}[OF \langle \text{prime } p \rangle]$ 
    by  $(\text{cases } p = 2) \text{ auto}$ 

  have  $\text{odd } p$  using  $\text{prime-odd-int}[OF \langle \text{prime } p \rangle \langle 2 < p \rangle]$  .

  have  $0 < x$ 
    using  $\langle 1 < (p * x) \rangle \text{prime-gt-0-int}[OF \langle \text{prime } p \rangle]$ 
    and  $\text{less-trans less-numeral-extra}(1) \text{ zero-less-mult-pos}$  by blast

```

```

have base-case : Jacobi 2 p = (- 1) ^ (((nat p)2 - 1) div 8)
using ⟨2 < p⟩ ⟨prime p⟩ supplement2-Legendre and prime-p-Jacobi-eq-Legendre
by presburger

show ?case proof (cases x = 1)
  case True
  thus ?thesis using base-case by force
next
  case False
  have Jacobi 2 (p * x) = Jacobi 2 p * Jacobi 2 x
    using ⟨2 < p⟩ ⟨0 < x⟩ by simp

  also have Jacobi 2 x = (- 1) ^ (((nat x)2 - 1) div 8)
    using ⟨odd x⟩ ⟨0 < x⟩ ⟨x ≠ 1⟩ by (intro factor.IH) auto

  also note base-case

  also have (-1) ^ (((nat p)2 - 1) div 8) * (-1) ^ (((nat x)2 - 1) div 8)
    = (-1 :: int) ^ (((nat (p * x))2 - 1) div 8)
    unfolding minus-one-power-iff
    using ⟨2 < p⟩ ⟨0 < x⟩ ⟨odd x⟩ ⟨odd p⟩ and odd-odd-even-nat'
    using [[linarith-split-limit = 0]]
    by (force simp add: nat-mult-distrib even-nat-iff)

  finally show ?thesis .
qed
qed simp-all

lemma mod-nat-wlog [consumes 1, case-names modulo]:
  fixes P :: nat ⇒ bool
  assumes b > 0
  assumes ∧k. k ∈ {0..<b} ⇒ n mod b = k ⇒ P n
  shows P n
  using assms and mod-less-divisor
  by fastforce

lemma mod-int-wlog [consumes 1, case-names modulo]:
  fixes P :: int ⇒ bool
  assumes b > 0
  assumes ∧k. 0 ≤ k ⇒ k < b ⇒ n mod b = k ⇒ P n
  shows P n
  using ⟨b > 0⟩ assms(2) [of ⟨n mod b⟩] by simp

lemma supplement2-Jacobi':
  assumes odd p and p > 1
  shows Jacobi 2 p = (if p mod 8 = 1 ∨ p mod 8 = 7 then 1 else -1)
proof -
  have 0 < (4 :: nat) by simp
  then have *: even ((p2 - 1) div 8) = (p mod 8 = 1 ∨ p mod 8 = 7) if odd p

```

```

for  $p :: \text{nat}$ 
  proof (induction  $p$  rule: mod-nat-wlog)
    case (modulo  $k$ )
      then consider  $p \bmod 4 = 1 \mid p \bmod 4 = 3$ 
        using  $\langle \text{odd } p \rangle$ 
        by (metis dvd-0-right even-even-mod-4-iff even-numeral mod-exhaust-less-4)

      then show ?case proof (cases)
        case 1
          then obtain  $l$  where  $l: p = 4 * l + 1$  using mod-natE by blast
          have even  $l = ((4 * l + 1) \bmod 8 = 1 \vee (4 * l + 1) \bmod 8 = 7)$  by
presburger
          thus ?thesis by (simp add: l power2-eq-square algebra-simps)
        next
          case 2
            then obtain  $l$  where  $l: p = 4 * l + 3$  using mod-natE by blast
            have odd  $l = ((3 + l * 4) \bmod 8 = \text{Suc } 0 \vee (3 + l * 4) \bmod 8 = 7)$  by
presburger
            thus ?thesis by (simp add: l power2-eq-square algebra-simps)
          qed
        qed

      have [simp]:  $\text{nat } p \bmod 8 = \text{nat } (p \bmod 8)$ 
        using  $\langle p > 1 \rangle$  using nat-mod-distrib[of  $p$  8] by simp
      from assms have odd (nat  $p$ ) by (simp add: even-nat-iff)
      show ?thesis
        unfolding supplement2-Jacobi[OF assms]
          minus-one-power-iff *[OF  $\langle \text{odd } (\text{nat } p) \rangle$ ]
        by (simp add: nat-eq-iff)
      qed

theorem supplement1-Jacobi:
  odd  $p \implies 1 < p \implies \text{Jacobi } (-1) p = (-1) ^ (\text{nat } ((p - 1) \text{ div } 2))$ 
proof (induction  $p$  rule: prime-divisors-induct)
  case (factor  $p x$ )
    then have odd  $x$  by force

    have  $2 < p$ 
      using  $\langle \text{odd } (p * x) \rangle$  prime-gt-1-int[OF  $\langle \text{prime } p \rangle$ ]
      by (cases  $p = 2$ ) auto

    have prime (nat  $p$ )
      using  $\langle \text{prime } p \rangle$  prime-int-nat-transfer
      by blast

    have Jacobi  $(-1) p = \text{Legendre } (-1) p$ 
      using prime-p-Jacobi-eq-Legendre[OF  $\langle \text{prime } p \rangle$ ] .

    also have ... =  $(-1) ^ ((\text{nat } p - 1) \text{ div } 2)$ 

```


using $\langle \text{prime } p \rangle \langle 2 < p \rangle$ **and** *supplement1-Legendre*[of nat p]
by (*metis int-nat-eq nat-mono-iff nat-numeral-as-int prime-gt-0-int prime-int-nat-transfer*)

also have $((\text{nat } p - 1) \text{ div } 2) = \text{nat } ((p - 1) \text{ div } 2)$ **by force**

finally have *base-case*: $\text{Jacobi } (-1) p = (-1) ^{\text{nat } ((p - 1) \text{ div } 2)}$.

show *?case proof* (*cases x = 1*)

case *True*

then show *?thesis using base-case by simp*

next

case *False*

have $0 < x$

using $\langle 1 < (p * x) \rangle$ *prime-gt-0-int*[*OF* $\langle \text{prime } p \rangle$]

by (*meson int-one-le-iff-zero-less not-less not-less-iff-gr-or-eq zero-less-mult-iff*)

have *odd p using* $\langle \text{prime } p \rangle \langle 2 < p \rangle$ **by** (*simp add: prime-odd-int*)

have $\text{Jacobi } (-1) (p * x) = \text{Jacobi } (-1) p * \text{Jacobi } (-1) x$

using $\langle 2 < p \rangle \langle 0 < x \rangle$ **by** *simp*

also note *base-case*

also have $\text{Jacobi } (-1) x = (-1) ^{\text{nat } ((x - 1) \text{ div } 2)}$

using $\langle 0 < x \rangle$ *False* $\langle \text{odd } x \rangle$ *factor.IH*

by *fastforce*

also have $(-1) ^{\text{nat } ((p - 1) \text{ div } 2)} * (-1) ^{\text{nat } ((x - 1) \text{ div } 2)} =$
 $(-1 :: \text{int}) ^{\text{nat } ((p * x - 1) \text{ div } 2)}$

unfolding *minus-one-power-iff*

using $\langle 2 < p \rangle \langle 0 < x \rangle$ **and** $\langle \text{odd } x \rangle \langle \text{odd } p \rangle$

by (*fastforce elim!: oddE simp: even-nat-iff algebra-simps*)

finally show *?thesis* .

qed

qed *simp-all*

theorem *supplement1-Jacobi'*:

$\text{odd } n \implies 1 < n \implies \text{Jacobi } (-1) n = (\text{if } n \bmod 4 = 1 \text{ then } 1 \text{ else } -1)$

by (*simp add: even-nat-iff minus-one-power-iff supplement1-Jacobi*)

presburger?

lemma *Jacobi-0-eq-0*: $\neg \text{is-unit } n \implies \text{Jacobi } 0 n = 0$

by (*cases prime-factorization n = {#}*)

(*auto simp: Jacobi-def prime-factorization-empty-iff image-iff intro: Nat.gr0I*)

lemma *is-unit-Jacobi-aux*: $\text{is-unit } x \implies \text{Jacobi } a x = 1$

unfolding *Jacobi-def using prime-factorization-empty-iff*[of x] **by** *auto*

lemma *is-unit-Jacobi*[simp]: $Jacobi\ a\ 1 = 1$ $Jacobi\ a\ (-1) = 1$
using *is-unit-Jacobi-aux* **by** *simp-all*

lemma *Jacobi-neg-right* [simp]:

$Jacobi\ a\ (-n) = Jacobi\ a\ n$

proof –

have * : $-n = (-1) * n$ **by** *simp*

show *?thesis* **unfolding** *

by (*subst Jacobi-mult-right*) *auto*

qed

lemma *Jacobi-neg-left*:

assumes *odd n 1 < n*

shows $Jacobi\ (-a)\ n = (if\ n\ mod\ 4 = 1\ then\ 1\ else\ -1) * Jacobi\ a\ n$

proof –

have * : $-a = (-1) * a$ **by** *simp*

show *?thesis* **unfolding** * *Jacobi-mult-left supplement1-Jacobi'*[*OF assms*] ..

qed

function *jacobi-code* :: $int \Rightarrow int \Rightarrow int$ **where**

jacobi-code $a\ n = ($

$\quad if\ n = 0\ then\ 0$

$\quad else\ if\ n = 1\ then\ 1$

$\quad else\ if\ a = 1\ then\ 1$

$\quad else\ if\ n < 0\ then\ jacobi-code\ a\ (-n)$

$\quad else\ if\ even\ n\ then\ if\ even\ a\ then\ 0\ else\ jacobi-code\ a\ (n\ div\ 2)$

$\quad else\ if\ a < 0\ then\ (if\ n\ mod\ 4 = 1\ then\ 1\ else\ -1) * jacobi-code\ (-a)\ n$

$\quad else\ if\ a = 0\ then\ 0$

$\quad else\ if\ a \geq n\ then\ jacobi-code\ (a\ mod\ n)\ n$

$\quad else\ if\ even\ a\ \ \ \ \ \ then\ (if\ n\ mod\ 8 \in \{1, 7\}\ then\ 1\ else\ -1) * jacobi-code\ (a$

$\quad div\ 2)\ n$

$\quad else\ if\ coprime\ a\ n\ then\ (if\ n\ mod\ 4 = 3 \wedge a\ mod\ 4 = 3\ then\ -1\ else\ 1) *$

$\quad jacobi-code\ n\ a$

$\quad else\ 0)$

by *auto*

termination

proof (*relation measure* $(\lambda(a, n). nat(abs(a) + abs(n)*2) +$

$(if\ n < 0\ then\ 1\ else\ 0) + (if\ a < 0\ then\ 1\ else\ 0)), goal-cases)$

case $(5\ a\ n)$

thus *?case* **by** (*fastforce intro!*: *less-le-trans*[*OF pos-mod-bound*])

qed *auto*

lemmas [*simp del*] = *jacobi-code.simps*

lemma *Jacobi-code* [*code*]: $Jacobi\ a\ n = jacobi-code\ a\ n$

proof (*induction a n rule: jacobi-code.induct*)

case $(1\ a\ n)$

show *?case*

```

proof (cases n = 0)
  case 2: False
  then show ?thesis proof (cases n = 1)
    case 3: False
    then show ?thesis proof (cases a = 1)
      case 4: False
      then show ?thesis proof (cases n < 0)
        case True
        then show ?thesis using 2 3 4 1(1) by (subst jacobi-code.simps) simp
        next
        case 5: False
        then show ?thesis proof (cases even n)
          case True
          then show ?thesis using 2 3 4 5 1(2)
          by (elim evenE, subst jacobi-code.simps) (auto simp: prime-p-Jacobi-eq-Legendre)
          next
          case 6: False
          then show ?thesis proof (cases a < 0)
            case True
            then show ?thesis using 2 3 4 5 6
            by (subst jacobi-code.simps, subst 1(3)[symmetric]) (simp-all add:
Jacobi-neg-left)
            next
            case 7: False
            then show ?thesis proof (cases a = 0)
              case True
              have *:  $\neg$  is-unit n using 3 5 by simp
              then show ?thesis
                using Jacobi-0-eq-0[OF *] 2 3 4 5 7 True
                by (subst jacobi-code.simps) simp
              next
              case 8: False
              then show ?thesis proof (cases a  $\geq$  n)
                case True
                then show ?thesis using 2 3 4 5 6 7 8 1(4)
                by (subst jacobi-code.simps) simp
                next
                case 9: False
                then show ?thesis proof (cases even a)
                  case True
                  hence a = 2 * (a div 2) by simp
                  also have Jacobi ... n = Jacobi 2 n * Jacobi (a div 2) n
                    by simp
                  also have Jacobi (a div 2) n = jacobi-code (a div 2) n
                    using 2 3 4 5 6 7 8 9 True by (intro 1(5))
                  also have Jacobi 2 n = (if n mod 8  $\in$  {1, 7} then 1 else - 1)
                    using 2 3 5 supplement2-Jacobi'[OF 6] by simp
                  also have ... * jacobi-code (a div 2) n = jacobi-code a n
                    using 2 3 4 5 6 7 8 9 True

```

```

    by (subst (2) jacobi-code.simps) (simp only: if-False if-True
HOL.simp-thms)
  finally show ?thesis .
next
case 10: False
note foo = 1 2 3
then show ?thesis proof (cases coprime a n)
  case True
  note this-case = 2 3 4 5 6 7 8 9 10 True
  have 2 < a using 10 4 7 by presburger
  moreover have 2 < n using 3 5 6 by presburger
  ultimately have jacobi-code a n = (if n mod 4 = 3 ∧ a mod
4 = 3 then - 1 else 1)
    * jacobi-code n a
  using this-case by (subst jacobi-code.simps) simp
  also have jacobi-code n a = Jacobi n a
  using this-case by (intro 1(6) [symmetric]) auto
  also have (if n mod 4 = 3 ∧ a mod 4 = 3 then -1 else 1) *
... = Jacobi a n
    using this-case and ⟨2 < a⟩
    by (intro Quadratic-Reciprocity-Jacobi' [symmetric])
      (auto simp: coprime-commute)
  finally show ?thesis ..
next
case False
have *: 0 < a 0 < n using 5 7 8 9 by linarith+
show ?thesis
  using 1 2 3 4 5 6 7 8 9 10 False *
by (subst jacobi-code.simps) (auto simp: Jacobi-eq-0-not-coprime)
qed
qed
qed
qed
qed
qed
qed (subst jacobi-code.simps, simp)
qed (subst jacobi-code.simps, simp)
qed (subst jacobi-code.simps, simp)
qed

```

lemma *Jacobi-eq-0-imp-not-coprime*:
 assumes $p \neq 0$ $p \neq 1$
 shows $Jacobi\ n\ p = 0 \implies \neg coprime\ n\ p$
 using *assms Jacobi-mod-cong coprime-iff-invertible-int* by force

lemma *Jacobi-eq-0-iff-not-coprime*:
 assumes $p \neq 0$ $p \neq 1$
 shows $Jacobi\ n\ p = 0 \iff \neg coprime\ n\ p$

```

proof –
  from assms and Jacobi-eq-0-imp-not-coprime
  show ?thesis using Jacobi-eq-0-not-coprime by auto
qed

end

```

4 Residue Rings of Natural Numbers

```

theory Residues-Nat
  imports Algebraic-Auxiliaries
begin

```

4.1 The multiplicative group of residues modulo n

```

definition Residues-Mult :: 'a :: {linordered-semidom, euclidean-semiring}  $\Rightarrow$  'a
  monoid where
    Residues-Mult  $p =$ 
      (carrier = { $x \in \{1..p\} . \text{coprime } x \ p$ }, monoid.mult =  $\lambda x \ y. x * y \bmod p$ , one
      = 1)

```

```

locale residues-mult-nat =
  fixes  $n :: \text{nat}$  and  $G$ 
  assumes n-gt-1:  $n > 1$ 
  defines  $G \equiv \text{Residues-Mult } n$ 
begin

```

```

lemma carrier-eq [simp]: carrier  $G = \text{totatives } n$ 
and mult-eq [simp]:  $(x \otimes_G y) = (x * y) \bmod n$ 
and one-eq [simp]:  $\mathbf{1}_G = 1$ 
by (auto simp: G-def Residues-Mult-def totatives-def)

```

```

lemma mult-eq':  $(\otimes_G) = (\lambda x \ y. (x * y) \bmod n)$ 
by (intro ext; simp)+

```

```

sublocale group  $G$ 
proof(rule groupI, goal-cases)
  case ( $1 \ x \ y$ )
  from 1 show ?case using n-gt-1
  by (auto intro!: Nat.gr0I simp: coprime-commute coprime-dvd-mult-left-iff
      coprime-absorb-left nat-dvd-not-less totatives-def)

```

```

next
  case ( $5 \ x$ )
  hence  $(\exists y. y \geq 0 \wedge y < n \wedge [x * y = \text{Suc } 0] \pmod n)$ 
  using coprime-iff-invertible'-nat[of n x] n-gt-1
  by (auto simp: totatives-def)
  then obtain  $y$  where  $y: y \geq 0 \ y < n \ [x * y = \text{Suc } 0] \pmod n$  by blast

  from  $\langle [x * y = \text{Suc } 0] \pmod n \rangle$  have  $\text{gcd } (x * y) \ n = 1$ 

```

```

    by (simp add: cong-gcd-eq)
  hence coprime y n by fastforce

  with y n-gt-1 show  $\exists y \in \text{carrier } G. y \otimes_G x = \mathbf{1}_G$ 
    by (intro bexI[of - y]) (auto simp: totatives-def cong-def mult-ac intro!: Nat.grOI)
qed (use n-gt-1 in ⟨auto simp: mod-simps algebra-simps totatives-less⟩)

sublocale comm-group
  by unfold-locales (auto simp: mult-ac)

lemma nat-pow-eq [simp]:  $x [\frown]_G (k :: \text{nat}) = (x \wedge k) \text{ mod } n$ 
  using n-gt-1 by (induction k) (simp-all add: mod-mult-left-eq mod-mult-right-eq
  mult-ac)

lemma nat-pow-eq':  $([\frown]_G) = (\lambda x k. (x \wedge k) \text{ mod } n)$ 
  by (intro ext) simp

lemma order-eq:  $\text{order } G = \text{totient } n$ 
  by (simp add: order-def totient-def)

lemma order-less:  $\neg \text{prime } n \implies \text{order } G < n - 1$ 
  using totient-less-not-prime[of n] n-gt-1
  by (auto simp: order-eq)

lemma ord-residue-mult-group:
  assumes  $a \in \text{totatives } n$ 
  shows  $\text{local.ord } a = \text{Pocklington.ord } n a$ 
proof (rule dvd-antisym)
  have  $[a \wedge \text{local.ord } a = 1] \text{ (mod } n)$ 
    using pow-ord-eq-1[of a] assms by (auto simp: cong-def)
  thus  $\text{Pocklington.ord } n a \text{ dvd local.ord } a$ 
    by (subst (asm) ord-divides)
next
  show  $\text{local.ord } a \text{ dvd Pocklington.ord } n a$ 
    using assms Pocklington.ord[of a n] n-gt-1 pow-eq-id by (simp add: cong-def)
qed

end

```

4.2 The ring of residues modulo n

definition *Residues-nat* :: $\text{nat} \Rightarrow \text{nat ring}$ where

Residues-nat $m = (\text{carrier} = \{0..<m\}, \text{monoid.mult} = \lambda x y. (x * y) \text{ mod } m, \text{one} = 1,$

$\text{ring.zero} = 0, \text{add} = \lambda x y. (x + y) \text{ mod } m)$

locale *residues-nat* =
 fixes $n :: \text{nat}$ and R
 assumes *n-gt-1*: $n > 1$

```

defines  $R \equiv \text{Residues-nat } n$ 
begin

lemma carrier-eq [simp]:  $\text{carrier } R = \{0..<n\}$ 
and mult-eq [simp]:  $x \otimes_R y = (x * y) \bmod n$ 
and add-eq [simp]:  $x \oplus_R y = (x + y) \bmod n$ 
and one-eq [simp]:  $\mathbf{1}_R = 1$ 
and zero-eq [simp]:  $\mathbf{0}_R = 0$ 
by (simp-all add: Residues-nat-def R-def)

lemma mult-eq':  $(\otimes_R) = (\lambda x y. (x * y) \bmod n)$ 
and add-eq':  $(\oplus_R) = (\lambda x y. (x + y) \bmod n)$ 
by (intro ext; simp)+

sublocale abelian-group  $R$ 
proof(rule abelian-groupI, goal-cases)
  case ( $1\ x\ y$ )
  then show ?case
    using n-gt-1
    by (auto simp: mod-simps algebra-simps simp flip: less-Suc-eq-le)
next
  case ( $6\ x$ )
  { assume  $x < n\ 1 < n$ 
    hence  $n - x \in \{0..<n\}\ ((n - x) + x) \bmod n = 0$  if  $x \neq 0$ 
    using that by auto
    moreover have  $0 \in \{0..<n\}\ (0 + x) \bmod n = 0$  if  $x = 0$ 
    using that n-gt-1 by auto
    ultimately have  $\exists y \in \{0..<n\}. (y + x) \bmod n = 0$ 
    by meson
  }

  with  $6$  show ?case using n-gt-1 by auto
qed (use n-gt-1 in <auto simp add: mod-simps algebra-simps>)

sublocale comm-monoid  $R$ 
using n-gt-1 by unfold-locales (auto simp: mult-ac mod-simps)

sublocale cring  $R$ 
by unfold-locales (auto simp: mod-simps algebra-simps)

lemma Units-eq:  $\text{Units } R = \text{totatives } n$ 
proof safe
  fix  $x$  assume  $x \in \text{Units } R$ 
  then obtain  $y$  where  $[x * y = 1] \pmod n$ 
  using n-gt-1 by (auto simp: Units-def cong-def)
  hence coprime  $x\ n$ 
  using cong-imp-coprime cong-sym coprime-1-left coprime-mult-left-iff by metis
  with  $x$  show  $x \in \text{totatives } n$  by (auto simp: totatives-def Units-def intro!:
Nat.gr0I)

```

```

next
  fix x assume x: x ∈ totatives n
  then obtain y where y < n [x * y = 1] (mod n)
    using coprime-iff-invertible'-nat[of n x] by (auto simp: totatives-def)
  with x show x ∈ Units R
    using n-gt-1 by (auto simp: Units-def mult-ac cong-def totatives-less)
qed

sublocale units: residues-mult-nat n units-of R
proof unfold-locales
  show units-of R ≡ Residues-Mult n
    by (auto simp: units-of-def Units-eq Residues-Mult-def totatives-def Suc-le-eq
mult-eq')
qed (use n-gt-1 in auto)

lemma nat-pow-eq [simp]: x [^]_R (k :: nat) = (x ^ k) mod n
  using n-gt-1 by (induction k) (auto simp: mod-simps mult-ac)

lemma nat-pow-eq': ([^]_R) = (λx k. (x ^ k) mod n)
  by (intro ext) simp

end

```

4.3 The ring of residues modulo a prime

```

locale residues-nat-prime =
  fixes p :: nat and R
  assumes prime-p: prime p
  defines R ≡ Residues-nat p
begin

sublocale residues-nat p R
  using prime-gt-1-nat[OF prime-p] by unfold-locales (auto simp: R-def)

lemma carrier-eq' [simp]: totatives p = {0 <..<p}
  using prime-p by (auto simp: totatives-prime)

lemma order-eq: order (units-of R) = p - 1
  using prime-p by (simp add: units.order-eq totient-prime)

lemma order-eq' [simp]: totient p = p - 1
  using prime-p by (auto simp: totient-prime)

sublocale field R
proof (rule cring-fieldI)
  show Units R = carrier R - {0_R}
    by (subst Units-eq) (use prime-p in ⟨auto simp: totatives-prime⟩)
qed

```



```

lemma residues-prime-cyclic:  $\exists x \in \{0 < .. < p\}. \{0 < .. < p\} = \{y. \exists i. y = x ^ i \text{ mod } p\}$ 
proof –
  from n-gt-1 have  $\{0 < .. < p\} - \{0\} = \{0 < .. < p\}$  by auto
  thus ?thesis using finite-field-mult-group-has-gen by simp
qed

lemma residues-prime-cyclic':  $\exists x \in \{0 < .. < p\}. \text{units.ord } x = p - 1$ 
proof –
  from residues-prime-cyclic obtain x
  where  $x: x \in \{0 < .. < p\} \{0 < .. < p\} = \{y. \exists i. y = x ^ i \text{ mod } p\}$  by metis
  have  $\text{units.ord } x = p - 1$ 
  proof (intro antisym)
    show  $\text{units.ord } x \leq p - 1$ 
    using units.ord-dvd-group-order[of x] x(1) by (auto simp: units.order-eq intro!: dvd-imp-le)
  next

  have  $p - 1 = \text{card } \{0 < .. < p\}$  by simp
  also have  $\{0 < .. < p\} = \{y. \exists i. y = x ^ i \text{ mod } p\}$  by fact
  also have  $\text{card } \dots \leq \text{card } ((\lambda i. x ^ i \text{ mod } p) ' \{.. < \text{units.ord } x\})$ 
  proof (intro card-mono; safe?)
    fix j :: nat
    have  $j = \text{units.ord } x * (j \text{ div } \text{units.ord } x) + (j \text{ mod } \text{units.ord } x)$ 
    by simp
    also have  $x [\bigwedge_{\text{units-of } R} \dots = x [\bigwedge_{\text{units-of } R} (\text{units.ord } x * (j \text{ div } \text{units.ord } x)$ 
x))
       $\otimes_{\text{units-of } R} x [\bigwedge_{\text{units-of } R} (j \text{ mod } \text{units.ord } x)$ 
    using x by (subst units.nat-pow-mult) auto
    also have  $x [\bigwedge_{\text{units-of } R} (\text{units.ord } x * (j \text{ div } \text{units.ord } x)) =$ 
       $(x [\bigwedge_{\text{units-of } R} \text{units.ord } x) [\bigwedge_{\text{units-of } R} (j \text{ div } \text{units.ord } x)$ 
    using x by (subst units.nat-pow-pow) auto
    also have  $x [\bigwedge_{\text{units-of } R} \text{units.ord } x = 1$ 
    using x(1) by (subst units.pow-ord-eq-1) auto
    finally have  $x ^ j \text{ mod } p = x ^ (j \text{ mod } \text{units.ord } x) \text{ mod } p$  using n-gt-1 by
simp
    thus  $x ^ j \text{ mod } p \in (\lambda i. x ^ i \text{ mod } p) ' \{.. < \text{units.ord } x\}$ 
    using units.ord-ge-1[of x] x(1) by force
  qed auto
  also have  $\dots \leq \text{card } \{.. < \text{units.ord } x\}$ 
  by (intro card-image-le) auto
  also have  $\dots = \text{units.ord } x$  by simp
  finally show  $p - 1 \leq \text{units.ord } x$  .
qed
with x show ?thesis by metis
qed
end

```

4.4 -1 in residue rings

lemma *minus-one-cong-solve-weak*:

fixes $n x :: \text{nat}$
assumes $1 < n$ $x \in \text{totatives } n$ $y \in \text{totatives } n$
and $[x = n - 1] \pmod{n}$ $[x * y = 1] \pmod{n}$
shows $y = n - 1$

proof –

define G **where** $G = \text{Residues-Mult } n$

interpret *residues-mult-nat* $n G$

by *unfold-locales* (*use* $\langle n > 1 \rangle$) **in** $\langle \text{simp-all add: } G\text{-def} \rangle$

have $[x * (n - 1) = x * n - x] \pmod{n}$

by (*simp add: algebra-simps*)

also have $[x * n - x = (n - 1) * n - (n - 1)] \pmod{n}$

using *assms* **by** (*intro cong-diff-nat cong-mult*) *auto*

also have $(n - 1) * n - (n - 1) = (n - 1) ^ 2$

by (*simp add: power2-eq-square algebra-simps*)

also have $[(n - 1)^2 = 1] \pmod{n}$

using *assms* **by** (*intro square-minus-one-cong-one*) *auto*

finally have $x * (n - 1) \pmod{n} = 1$

using $\langle n > 1 \rangle$ **by** (*simp add: cong-def*)

hence $y = n - 1$

using *inv-unique'*[*of* x $n - 1$] *inv-unique'*[*of* x y] *minus-one-in-totatives*[*of* n]

assms($1-3,5$)

by (*simp-all add: mult-ac cong-def*)

then show *?thesis* **by** *simp*

qed

lemma *coprime-imp-mod-not-zero*:

fixes $n x :: \text{nat}$

assumes $1 < n$ *coprime* x n

shows $0 < x \pmod{n}$

using *assms* *coprime-0-left-iff nat-dvd-not-less* **by** *fastforce*

lemma *minus-one-cong-solve*:

fixes $n x :: \text{nat}$

assumes $1 < n$

and *eq*: $[x = n - 1] \pmod{n}$ $[x * y = 1] \pmod{n}$

and *coprime*: *coprime* x n *coprime* y n

shows $[y = n - 1] \pmod{n}$

proof –

have $0 < x \pmod{n}$ $0 < y \pmod{n}$

using *coprime* *coprime-imp-mod-not-zero* $\langle 1 < n \rangle$ **by** *blast+*

moreover have $x \pmod{n} < n$ $y \pmod{n} < n$

using $\langle 1 < n \rangle$ **by** *auto*

moreover have $[x \pmod{n} = n - 1] \pmod{n}$ $[x \pmod{n} * (y \pmod{n}) = 1] \pmod{n}$

using *eq* **by** *auto*

moreover have *coprime* $(x \pmod{n})$ n *coprime* $(y \pmod{n})$ n

using *coprime* *coprime-mod-left-iff* $\langle 1 < n \rangle$ **by** *auto*

ultimately have $[y \pmod{n} = n - 1] \pmod{n}$

```

    using minus-one-cong-solve-weak[OF <1 < n>, of x mod n y mod n]
    by (auto simp: totatives-def)
  then show ?thesis by simp
qed

```

```

corollary square-minus-one-cong-one':
  fixes n x :: nat
  assumes 1 < n
  shows [(n - 1) * (n - 1) = 1](mod n)
  using square-minus-one-cong-one[OF assms, of n - 1] assms
  by (fastforce simp: power2-eq-square)

end

```

5 Additional Material on Quadratic Residues

```

theory QuadRes

```

```

imports

```

```

  Jacobi-Symbol

```

```

  Algebraic-Auxiliaries

```

```

begin

```

Proofs are inspired by [5].

```

lemma inj-on-QuadRes:

```

```

  fixes p :: int

```

```

  assumes prime p

```

```

  shows inj-on (λx. x2 mod p) {0..(p-1) div 2}

```

```

proof

```

```

  fix x y :: int

```

```

  assume elem: x ∈ {0..(p-1) div 2} y ∈ {0..(p-1) div 2}

```

```

  have * : abs(a) < p ⇒ p dvd a ⇒ a = 0 for a :: int

```

```

    using dvd-imp-le-int by force

```

```

  assume x2 mod p = y2 mod p

```

```

  hence [x2 = y2] (mod p) unfolding cong-def .

```

```

  hence p dvd (x2 - y2) by (simp add: cong-iff-dvd-diff)

```

```

  hence p dvd (x + y) * (x - y)

```

```

    by (simp add: power2-eq-square square-diff-square-factored)

```

```

  hence p dvd (x + y) ∨ p dvd (x - y)

```

```

    using <prime p> by (simp add: prime-dvd-mult-iff)

```

```

  moreover have p dvd x + y ⇒ x + y = 0 p dvd x - y ⇒ x - y = 0

```

```

    and 0 ≤ x 0 ≤ y

```

```

    using elem

```

by (*fastforce intro!*: *)+

ultimately show $x = y$ by *auto*
 qed

lemma *QuadRes-set-prime*:
 assumes *prime p* and *odd p*
 shows $\{x . \text{QuadRes } p \ x \wedge x \in \{0..<p\}\} = \{x^2 \bmod p \mid x . x \in \{0..(p-1) \text{ div } 2\}\}$

proof (*safe, goal-cases*)
 case (*1 x*)
 then obtain *y* where $[y^2 = x] \pmod{p}$
 unfolding *QuadRes-def* by *blast*

then have *A*: $[(y \bmod p)^2 = x] \pmod{p}$
 unfolding *cong-def*
 by (*simp add: power-mod*)

then have $[(-(y \bmod p))^2 = x] \pmod{p}$
 by *simp*

then have *B*: $[(p - (y \bmod p))^2 = x] \pmod{p}$
 unfolding *cong-def*
 using *minus-mod-self1*
 by (*metis power-mod*)

have $p = 1 + ((p - 1) \text{ div } 2) * 2$
 using *prime-gt-0-int[OF <prime p>] <odd p>*
 by *simp*

then have *C*: $(p - (y \bmod p)) \in \{0..(p - 1) \text{ div } 2\} \vee y \bmod p \in \{0..(p - 1) \text{ div } 2\}$
 using *prime-gt-0-int[OF <prime p>]*
 by (*clarsimp, auto simp: le-less*)

then show *?case proof*
 show *?thesis* if $p - y \bmod p \in \{0..(p - 1) \text{ div } 2\}$
 using *that B*
 unfolding *cong-def*
 using $\langle x \in \{0..<p\} \rangle$ by *auto*

show *?thesis* if $y \bmod p \in \{0..(p - 1) \text{ div } 2\}$
 using *that A*
 unfolding *cong-def*
 using $\langle x \in \{0..<p\} \rangle$ by *auto*

qed

qed (*auto simp: QuadRes-def cong-def*)

corollary *QuadRes-iff*:

assumes *prime p and odd p*
shows $(\text{QuadRes } p \ x \wedge x \in \{0..<p\}) \longleftrightarrow (\exists a \in \{0..(p-1) \text{ div } 2\}. a^2 \text{ mod } p = x)$
proof –
have $(\text{QuadRes } p \ x \wedge x \in \{0..<p\}) \longleftrightarrow x \in \{x. \text{QuadRes } p \ x \wedge x \in \{0..<p\}\}$
by *auto*
also note *QuadRes-set-prime[OF assms]*
also have $(x \in \{x^2 \text{ mod } p \mid x. x \in \{0..(p-1) \text{ div } 2\}\}) = (\exists a \in \{0..(p-1) \text{ div } 2\}. a^2 \text{ mod } p = x)$
by *blast*
finally show *?thesis .*
qed

corollary *card-QuadRes-set-prime:*

fixes $p :: \text{int}$
assumes *prime p and odd p*
shows $\text{card } \{x. \text{QuadRes } p \ x \wedge x \in \{0..<p\}\} = \text{nat } (p+1) \text{ div } 2$
proof –
have $\text{card } \{x. \text{QuadRes } p \ x \wedge x \in \{0..<p\}\} = \text{card } \{x^2 \text{ mod } p \mid x. x \in \{0..(p-1) \text{ div } 2\}\}$
unfolding *QuadRes-set-prime[OF assms] ..*

also have $\{x^2 \text{ mod } p \mid x. x \in \{0..(p-1) \text{ div } 2\}\} = (\lambda x. x^2 \text{ mod } p) ` \{0..(p-1) \text{ div } 2\}$
by *auto*

also have $\text{card } \dots = \text{card } \{0..(p-1) \text{ div } 2\}$
using *inj-on-QuadRes[OF <prime p>] by (rule card-image)*

also have $\dots = \text{nat } (p+1) \text{ div } 2$ **by** *simp*

finally show *?thesis .*
qed

corollary *card-not-QuadRes-set-prime:*

fixes $p :: \text{int}$
assumes *prime p and odd p*
shows $\text{card } \{x. \neg \text{QuadRes } p \ x \wedge x \in \{0..<p\}\} = \text{nat } (p-1) \text{ div } 2$
proof –
have $\{0..<p\} \cap \{x. \text{QuadRes } p \ x \wedge x \in \{0..<p\}\} = \{x. \text{QuadRes } p \ x \wedge x \in \{0..<p\}\}$
by *blast*

moreover have $\text{nat } p - \text{nat } (p+1) \text{ div } 2 = \text{nat } (p-1) \text{ div } 2$
using *<odd p> prime-gt-0-int[OF <prime p>]*
by *(auto elim!: oddE simp: nat-add-distrib nat-mult-distrib)*

ultimately have $\text{card } \{0..<p\} - \text{card } (\{0..<p\} \cap \{x. \text{QuadRes } p \ x \wedge x \in \{0..<p\}\}) = \text{nat } (p-1) \text{ div } 2$

using *card-QuadRes-set-prime*[*OF assms*] **and** *card-atLeastZeroLessThan-int*
by *presburger*

moreover have $\{x. \neg \text{QuadRes } p \ x \wedge x \in \{0..<p\}\} = \{0..<p\} - \{x. \text{QuadRes } p \ x \wedge x \in \{0..<p\}\}$
by *blast*

ultimately show *?thesis* **by** (*auto simp add: card-Diff-subset-Int*)
qed

lemma *not-QuadRes-ex-if-prime*:

assumes *prime p* **and** *odd p*

shows $\exists x. \neg \text{QuadRes } p \ x$

proof –

have $2 < p$ **using** *odd-prime-gt-2-int assms* **by** *blast*

then have *False* **if** $\{x. \neg \text{QuadRes } p \ x \wedge x \in \{0..<p\}\} = \{\}$

using *card-not-QuadRes-set-prime*[*OF assms*]

unfolding *that*

by *simp*

thus *?thesis* **by** *blast*

qed

lemma *not-QuadRes-ex*:

$1 < p \implies \text{odd } p \implies \exists x. \neg \text{QuadRes } p \ x$

proof (*induction p rule: prime-divisors-induct*)

case (*factor p x*)

then show *?case*

by (*meson not-QuadRes-ex-if-prime QuadRes-def cong-iff-dvd-diff dvd-mult-left even-mult-iff*)

qed *simp-all*

end

6 Euler Witnesses

theory *Euler-Witness*

imports

Jacobi-Symbol

Residues-Nat

QuadRes

begin

Proofs are inspired by [13, 8, 15, 11].

definition *euler-witness* $a \ p \longleftrightarrow [\text{Jacobi } a \ p \neq a^{\wedge((p-1) \text{ div } 2)}] \pmod p$

abbreviation *euler-liar* $a \ p \equiv \neg \text{euler-witness } a \ p$

lemma *euler-witness-mod*[*simp*]: $\text{euler-witness } (a \ \text{mod } p) \ p = \text{euler-witness } a \ p$

unfolding *euler-witness-def cong-def*
by (*simp add: power-mod*)

lemma *euler-liar-mod: euler-liar (a mod p) p = euler-liar a p by simp*

lemma *euler-liar-cong:*
assumes $[a = b] \pmod{p}$
shows *euler-liar a p = euler-liar b p*
by (*metis assms cong-def euler-witness-mod*)

lemma *euler-witnessI:*
 $[x^{((n-1) \text{ div } 2)} = a] \pmod{\text{int } n} \implies [\text{Jacobi } x \text{ (int } n) \neq a] \pmod{\text{int } n}$
 $\implies \text{euler-witness } x \ n$
unfolding *euler-witness-def using cong-trans by blast*

lemma *euler-witnessI2:*
fixes $a \ b :: \text{int}$ **and** $k :: \text{nat}$
assumes $[a \neq b] \pmod{k}$
and $k \text{ dvd } n$
and *main: euler-liar x n \implies [Jacobi x n = a] (mod k)*
 $\text{euler-liar } x \ n \implies [x^{((n-1) \text{ div } 2)} = b] \pmod{k}$
shows *euler-witness x n*
proof (*rule ccontr*)
assume *euler-liar x n*
then have $[\text{Jacobi } x \text{ (int } n) = x^{((n-1) \text{ div } 2)}] \pmod{k}$
using $\langle k \text{ dvd } n \rangle$ *cong-dvd-modulus euler-witness-def int-dvd-int-iff by blast*

moreover note *main[OF $\langle \text{euler-liar } x \ n \rangle$]*

ultimately show *False*
using $\langle [a \neq b] \pmod{k} \rangle$ **and** *cong-trans cong-sym*
by *metis*
qed

lemma *euler-witness-exists-weak:*
assumes $\text{odd } n \ \neg \text{prime } n \ 2 < n$
shows $\exists a. \text{euler-witness } a \ n \wedge \text{coprime } a \ n$
proof –
show *?thesis proof (cases squarefree n)*
case *True*

obtain $p \ k$ **where** $n = p * k$ **and** $1 < p \ p < n \ 1 < k \ k < n$ *prime p*
using *prime-divisor-exists-strong-nat[of n] $\langle \neg \text{prime } n \rangle \langle 2 < n \rangle$ by auto*

have *coprime p k using $\langle n = p * k \rangle$ and $\langle \text{squarefree } n \rangle$*
using *squarefree-mult-imp-coprime by blast*

hence *coprime (int p) (int k) by simp*

```

have odd p using n ⟨odd n⟩ by simp

then obtain b where ¬QuadRes p b
  using not-QuadRes-ex[of int p]
  using ⟨prime p⟩ prime-gt-1-int by auto

then have [b ≠ 0] (mod p)
  unfolding cong-def QuadRes-def
  by (metis zero-power2)

from binary-chinese-remainder-int[OF ⟨coprime (int p) (int k)⟩, of b 1]
obtain x :: int where x: [x = b] (mod p) [x = 1] (mod k) by blast

have euler-witness x n
proof (rule euler-witnessI2[of -1 1 k])
  show [x ^ ((n - 1) div 2) = 1] (mod k)
    using ⟨[x = 1] (mod k)⟩ and cong-def
    using cong-pow by fastforce
next
have Jacobi x n = Jacobi x p * Jacobi x k
  unfolding n
  using ⟨1 < k⟩ ⟨1 < p⟩ by fastforce

also note Jacobi-mod-cong[OF ⟨[x = b] (mod p)⟩]
also have Jacobi b p = Legendre b p
  using ⟨prime p⟩ by (simp add: prime-p-Jacobi-eq-Legendre)
also have ... = -1
  unfolding Legendre-def
  using ⟨[b ≠ 0] (mod p)⟩ and ⟨¬ QuadRes p b⟩ by auto

also note Jacobi-mod-cong[OF ⟨[x = 1] (mod k)⟩]

finally show [Jacobi x (int n) = - 1] (mod int k)
  using ⟨1 < k⟩ by auto
next
have 2 < k
  using ⟨odd n⟩ and ⟨1 < k⟩ unfolding n
  by(cases k = 2) auto

then show [- 1 ≠ 1] (mod k) by auto
next
show k dvd n unfolding n by simp
qed

have coprime x p
  using ⟨[b ≠ 0] (mod int p)⟩ ⟨[x = b] (mod int p)⟩ and ⟨prime p⟩ not-coprime-cong-eq-0[of
p x] prime-nat-int-transfer
  by (blast intro: cong-sym cong-trans)

```



```

then have coprime x n
  using n ⟨[x = 1] (mod int k)⟩
  by (metis coprime-iff-invertible-int coprime-mult-right-iff mult.right-neutral
of-nat-mult)

then show ?thesis
  using ⟨euler-witness x n⟩ by blast

next
case False
then obtain p where p: prime p p^2 dvd n using ⟨odd n⟩
  by (metis less-not-refl2 odd-pos squarefree-factorial-semiring)
hence p dvd n by auto

from p have Z: p dvd n div p by (auto intro: dvd-mult-imp-div simp: power2-eq-square)

have n: n = p * (n div p)
  using p(2) unfolding power2-eq-square by (metis dvd-mult-div-cancel dvd-mult-right)

have odd p using p ⟨odd n⟩
  by (meson dvd-trans even-power pos2)

then have 2 < p using prime-ge-2-nat[OF ⟨prime p⟩]
  by presburger

define a where a-def: a = n div p + 1

have A: a^p = (∑ k=0..p. (p choose k) * (n div p)^k)
  unfolding binomial a-def
  using atLeast0AtMost by auto

also have B: ... = 1 + (∑ k = 1..p. (p choose k) * (n div p)^k) (is ?A = 1
+ ?B)
  by (simp add: sum.atLeast-Suc-atMost)

also have C: ?B = (n div p) * (∑ k = 1..p. (p choose k) * (n div p)^(k -
1)) (is - = - * ?C)
  unfolding sum-distrib-left
  proof (rule sum.cong)
  fix x
  assume x ∈ {1..p}
  hence 0 < x by simp
  hence (n div p)^x = n div p * (n div p)^(x - 1)
    by (metis mult.commute power-minus-mult)
  thus (p choose x) * (n div p)^x = n div p * ((p choose x) * (n div p)^(
x-1))
    by simp
  qed simp

```

finally have $1: a \wedge p = 1 + n \text{ div } p * (\sum k = 1..p. (p \text{ choose } k) * (n \text{ div } p) \wedge (k - 1))$.

have $D: p \text{ dvd } ?C$
proof (*rule dvd-sum, goal-cases*)
 fix a
 assume $a \in \{1..p\}$
 show $p \text{ dvd } (p \text{ choose } a) * (n \text{ div } p) \wedge (a - 1)$
 proof(*cases a = p*)
 note $* = \text{dvd-power-le}[of - - 1, \text{simplified}]$
 case *True*
 thus *?thesis*
 using $\langle p \text{ dvd } n \text{ div } p \rangle \langle 2 < p \rangle$ **by** (*auto intro: **)
 next
 case *False*
 thus *?thesis*
 using $\langle a \in \{1..p\} \rangle$ **and** $\langle \text{prime } p \rangle$
 by (*auto intro!: dvd-mult2 prime-dvd-choose*)
 qed
qed

then obtain m **where** $m: ?C = p * m$
 using *dvdE* **by** *blast*

have $0 < p$ **using** $\langle \text{odd } p \rangle$ **and** *odd-pos* **by** *blast*

then have $?C \neq 0$
 using *sum.atLeast-Suc-atMost[OF Suc-leI]*
 by (*simp add: Suc-leI sum.atLeast-Suc-atMost*)

then have $m \neq 0$ **using** m **by** *fastforce*

have $n \text{ dvd } ?B$
 unfolding $C \ m$ **using** p **by** (*auto simp: power2-eq-square*)

hence $?B \text{ mod } n = 0$ **by** *presburger*

hence $[a \wedge p = 1] \text{ (mod } n)$
 unfolding $A \ B$ *cong-def*
 by (*subst mod-Suc-eq[symmetric, unfolded Suc-eq-plus1-left]*) *simp*

have $\neg p \text{ dvd } n - 1$
 using p *assms(1)*
 by (*metis One-nat-def Suc-leI dvd-diffD1 dvd-mult-right not-prime-unit odd-pos power2-eq-square*)

have $[(n \text{ div } p + 1) \neq 1] \text{ (mod } n)$
 using $\langle 2 < p \rangle \langle \text{odd } n \rangle$ **and** $\langle \text{prime } p \rangle \langle p^2 \text{ dvd } n \rangle$
 using *div-mult2-eq n nonzero-mult-div-cancel-left* **by** (*force dest!: cong-to-1-nat*)

then have $\text{ord } n \ a \neq 1$
using $\langle 2 < p \rangle \langle \text{odd } n \rangle$ **and** $\langle \text{prime } p \rangle \langle p^2 \ \text{dvd } n \rangle$
using $\text{ord-works}[of \ a \ n]$
unfolding $a\text{-def}$
by auto

then have $\text{ord } n \ a = p$
using $\text{ord-divides}[of \ a \ p \ n] \langle [a \wedge p = 1] \ (\text{mod } n) \rangle \langle \text{prime } p \rangle$
using prime-nat-iff **by** blast

have $\text{coprime } n \ a$
using $\langle \text{prime } p \rangle \langle p^2 \ \text{dvd } n \rangle \langle p \ \text{dvd } n \ \text{div } p \rangle \ n$
unfolding $a\text{-def}$
by $(\text{metis coprime-add-one-right coprime-mult-left-iff dvd-def})$

have $[(n - 1) \ \text{div } 2 \neq 0] \ (\text{mod } p)$
using $\langle \neg \ p \ \text{dvd } n - 1 \rangle \langle \text{odd } n \rangle$
by $(\text{subst cong-altdef-nat}) \ (\text{auto elim!: oddE})$

then have $[p \neq (n - 1) \ \text{div } 2] \ (\text{mod } p)$
using $\text{cong-mult-self-right}[of \ 1 \ p, \ \text{simplified}] \ \text{cong-sym} \ \text{cong-trans}$ **by** blast

then have $[a \wedge ((n-1) \ \text{div } 2) \neq 1] \ (\text{mod } n)$
using $\langle [a \wedge p = 1] \ (\text{mod } n) \rangle$
using $\text{order-divides-expdiff}[OF \ \langle \text{coprime } n \ a \rangle, \ \text{of } p \ (n-1) \ \text{div } 2, \ \text{symmetric}]$
unfolding $\langle \text{ord } n \ a = p \rangle$
using $\text{cong-sym} \ \text{cong-trans}$
by blast

then have $[(\text{int } a) \wedge ((n-1) \ \text{div } 2) \neq 1] \ (\text{mod } n)$
unfolding cong-def
by $(\text{metis of-nat-1 of-nat-eq-iff of-nat-mod of-nat-power})$

have $\text{Jacobi } a \ n = \text{Jacobi } a \ (p * \text{int } (n \ \text{div } p))$
using n **by** $(\text{metis of-nat-mult})$

also have $\dots = \text{Jacobi } a \ p * \text{Jacobi } a \ (n \ \text{div } p)$
using $\langle \text{odd } n \rangle$ **and** $\langle n = p * (n \ \text{div } p) \rangle$
by $(\text{subst Jacobi-mult-right}) \ \text{auto}$

also have $\text{Jacobi } a \ p = \text{Jacobi } 1 \ p$
using $\langle p \ \text{dvd } n \ \text{div } p \rangle$
by $(\text{intro Jacobi-mod-cong}) \ (\text{auto simp: cong-iff-dvd-diff } a\text{-def})$

also have $\dots = 1$
by $(\text{simp add: } \langle 0 < p \rangle)$

also have $\text{Jacobi } a \ (n \ \text{div } p) = \text{Jacobi } 1 \ (n \ \text{div } p)$

by (rule Jacobi-mod-cong) (simp add: cong-iff-dvd-diff a-def)

also have ... = 1
 using ⟨p dvd n⟩ ⟨prime p⟩ ⟨n > 2⟩
 by (intro Jacobi-1-eq-1) (auto intro!: Nat.gr0I elim!: dvdE)

finally show ?thesis using ⟨[int a ^ ((n - 1) div 2) ≠ 1] (mod n)⟩ ⟨coprime n a⟩
 unfolding euler-witness-def
 by (intro exI[of - int a]) (auto simp: cong-sym coprime-commute)

qed

lemma euler-witness-exists:
 assumes odd n ¬prime n 2 < n
 shows ∃ a. euler-witness a n ∧ coprime a n ∧ 0 < a ∧ a < n
 proof -
 obtain a where a: euler-witness a n coprime a n
 using euler-witness-exists-weak assms by blast

then have euler-witness (a mod n) n coprime (a mod n) n
 using ⟨odd n⟩ by (simp-all add: odd-pos)

moreover have 0 < (a mod n)
 proof -
 have 0 ≤ (a mod n) by (rule pos-mod-sign) (use ⟨2 < n⟩ in simp)

moreover have 0 ≠ (a mod n)
 using ⟨coprime (a mod n) n⟩ coprime-0-left-iff[of int n] ⟨2 < n⟩ by auto

ultimately show ?thesis by linarith

qed

moreover have a mod n < n
 by (rule pos-mod-bound) (use ⟨2 < n⟩ in simp)

ultimately show ?thesis by blast

qed

lemma euler-witness-exists-nat:
 assumes odd n ¬prime n 2 < n
 shows ∃ a. euler-witness (int a) n ∧ coprime a n ∧ 0 < a ∧ a < n
 using euler-witness-exists[OF assms]
 using zero-less-imp-eq-int by fastforce

lemma euler-liar-1-p[intro, simp]: p ≠ 0 ⇒ euler-liar 1 p
 unfolding euler-witness-def by simp

lemma euler-liar-mult:

```

shows euler-liar  $y \ n \implies$  euler-liar  $x \ n \implies$  euler-liar  $(x*y) \ n$ 
unfolding euler-witness-def
by (auto simp: power-mult-distrib intro: cong-mult)

lemma euler-liar-mult':
  assumes  $1 < n$  coprime  $y \ n$ 
  shows euler-liar  $y \ n \implies$  euler-witness  $x \ n \implies$  euler-witness  $(x*y) \ n$ 
proof(simp add: euler-witness-def power-mult-distrib, rule cong-mult-uneq', goal-cases)
case 1
  then show ?case
    using Jacobi-eq-0-iff-not-coprime[ $of \ n \ y$ ] Jacobi-values[ $of \ y \ n$ ] and assms
    by auto
qed simp-all

lemma prime-imp-euler-liar:
  assumes prime  $p$   $2 < p$   $0 < x$   $x < p$ 
  shows euler-liar  $x \ p$ 
  using assms prime-p-Jacobi-eq-Legendre and euler-criterion
  unfolding euler-witness-def
  by simp

locale euler-witness-context =
  fixes  $p :: nat$ 
  assumes  $p-gt-1$ :  $p > 1$  and  $odd-p$ :  $odd \ p$ 
begin

definition  $G$  where  $G = Residues-Mult \ p$ 

sublocale residues-mult-nat  $p \ G$ 
  by unfold-locales (use  $p-gt-1$  in  $\langle simp-all \ add: \ G-def \rangle$ )

definition  $H = \{x. \ coprime \ x \ p \wedge \ euler-liar \ (int \ x) \ p \wedge \ x \in \{1..<p\}\}$ 

sublocale  $H$ : subgroup  $H \ G$ 
proof –
  have subset:  $H \subseteq carrier \ G$  by (auto simp:  $H-def \ totatives-def$ )
  show subgroup  $H \ G$ 
  proof (rule group.subgroupI, goal-cases)
    case 1
    then show ?case by (fact is-group)
  next
    case 3
    have euler-liar  $1 \ p$  using  $p-gt-1$ 
    unfolding euler-witness-def  $cong-def$  by simp
    then show ?case
    using  $p-gt-1$  by (auto simp:  $H-def$ )
  next
    case ( $\not\leq x$ )
    then have coprime  $x \ p$  euler-liar  $x \ p$   $1 \leq x$   $x < p$ 

```

```

    by (auto simp: H-def)

define y where y = inv_G x

from ⟨x ∈ H⟩ have x ⊗_G y = 1_G
  unfolding y-def using subset by (intro r-inv) auto

hence *: (x * y) mod p = 1 by (auto simp: y-def)
hence **: (int x * int y) mod p = 1
  by (metis of-nat-1 of-nat-mod of-nat-mult)

from * have coprime y p
  using p-gt-1 ⟨x < p⟩
  by (auto simp add: coprime-iff-invertible'-nat cong-def mult.commute)

moreover from 4 have y ∈ carrier G
  using subset unfolding y-def by (intro inv-closed) auto

hence 1 ≤ y y < p using p-gt-1 totatives-less[of y p]
  by (auto simp: totatives-def)

moreover have euler-liar 1 p
  using p-gt-1 by (intro euler-liar-1-p) auto
then have euler-liar (int x * int y) p
  using ** p-gt-1 by (subst euler-liar-cong[of int x * int y 1 p]) (auto simp:
cong-def)

then have euler-liar y p
  using ⟨coprime x p⟩ ⟨euler-liar x p⟩ and euler-liar-mult'[OF p-gt-1, of x y]
  by (metis coprime-int-iff mult.commute)

ultimately show ?case by (auto simp: H-def simp flip: y-def)
next
case (5 x y)
then show ?case
  unfolding euler-witness-def H-def
  by (auto intro!: gre1I-nat cong-mult
      simp: coprime-commute coprime-dvd-mult-left-iff
      nat-dvd-not-less zmod-int power-mult-distrib)
qed fact+
qed

lemma H-finite: finite H
  unfolding H-def by simp

lemma euler-witness-coset:
  assumes euler-witness x p
  shows y ∈ H #>_G x ⟹ euler-witness y p
  using assms euler-liar-mult'[OF p-gt-1, of - x] unfolding r-coset-is-image H-def

```

by (auto simp add: mult.commute of-nat-mod)

lemma *euler-liar-coset*:

assumes *euler-liar* $x p$ $x \in \text{carrier } G$

shows $y \in H \#>_G x \implies \text{euler-liar } y p$

using *is-group* *H.rcos-const*[of x] *assms* *totatives-less*[of $x p$] *p-gt-1*

by (auto simp: *H-def* *totatives-def*)

lemma *in-cosets-aux*:

assumes *euler-witness* $x p$ $x \in \text{carrier } G$

shows $H \#>_G x \in \text{rcosets}_G H$

using *assms* by (intro *rcosetsI*) (auto simp: *H-def* *totatives-def*)

lemma *H-cosets-aux*:

assumes *euler-witness* $x p$

shows $(H \#>_G x) \cap H = \{\}$

using *euler-witness-coset* *assms* **unfolding** *H-def* by **blast**

lemma *H-rcosets-aux*:

assumes *euler-witness* $x p$ $x \in \text{carrier } G$

shows $\{H, H \#>_G x\} \subseteq \text{rcosets}_G H$

using *in-cosets-aux*[OF *assms*] *H.subgroup-in-rcosets*[OF *is-group*]

by *blast*

lemma *H-not-eq-coset*:

assumes *euler-witness* $x p$

shows $H \neq H \#>_G x$

using *H.one-closed* *H-def* *assms*(1) *euler-witness-coset* by **blast**

lemma *finite-cosets-H*: *finite* (*rcosets* _{G} H)

using *rcosets-part-G*[OF *H.is-subgroup*]

by (auto intro: *finite-UnionD*)

lemma *card-cosets-limit*:

assumes *euler-witness* $x p$ $x \in \text{carrier } G$

shows $2 \leq \text{card } (\text{rcosets}_G H)$

using *H-not-eq-coset*[OF *assms*(1)] *card-mono*[OF *finite-cosets-H* *H-rcosets-aux*[OF *assms*]]

by *simp*

lemma *card-euler-liars-cosets-limit*:

assumes $\neg \text{prime } p$ $2 < p$

shows $2 \leq \text{card } (\text{rcosets}_G H)$ $\text{card } H < (p - 1) \text{ div } 2$

proof –

have $H \in \text{rcosets}_G H$ $H \subseteq \text{carrier } G$

by (auto simp: *is-group* *H.subgroup-in-rcosets* *simp* *del: carrier-eq*)

obtain $a :: \text{nat}$ **where** *euler-witness* $a p$ *coprime* $a p$ $0 < a < p$

using *odd-p assms euler-witness-exists-nat*
by *blast*

moreover have $a: a \in \text{carrier } G$
using *calculation* **by** (*auto simp: totatives-def*)

ultimately show $2 \leq \text{card } (\text{rcosets}_G H)$
using *card-cosets-limit* **by** *blast*

have *finite* H
by (*rule finite-subset*[*OF H.subset*]) *auto*

have *finite* $(H \#>_G a)$
by (*rule cosets-finite*[*OF rcosetsI*]) (*use H.subset a in auto*)

have $H \#>_G a \in \text{rcosets}_G H$
using *H.subset rcosetsI* $\langle a \in \text{carrier } G \rangle$ **by** *blast*

then have $\text{card } H = \text{card } (H \#>_G a)$
using *card-rcosets-equal H.subset* **by** *blast*

moreover have $H \cup H \#>_G a \subseteq \text{carrier } G$
using *rcosets-part-G*[*OF H.is-subgroup*]
using *H.subgroup-in-rcosets*[*OF is-group*] **and** $\langle H \#>_G a \in \text{rcosets}_G H \rangle$
by *auto*

then have $\text{card } H + \text{card } (H \#>_G a) \leq \text{card } (\text{carrier } G)$
using $\langle \text{finite } H \rangle \langle \text{finite } (H \#>_G a) \rangle$
using *H-cosets-aux*[*OF* $\langle \text{euler-witness } a \ p \rangle$]
using *H.subset finite-subset card-Un-disjoint*
by (*subst card-Un-disjoint*[*symmetric*]) (*auto intro: card-mono simp flip: card-Un-disjoint*)

ultimately have $\text{card } H \leq \text{card } (\text{carrier } G) \text{ div } 2$
by *linarith*

obtain $pa \ k$ **where** $pa: p = pa * k \ 1 < pa \ pa < p \ 1 < k \ k < p$ *prime* pa
using *prime-divisor-exists-strong-nat*[*OF p-gt-1* $\langle \neg \text{prime } p \rangle$]
by *blast*

hence $\neg \text{coprime } pa \ p$ **by** *simp*

then have $\text{carrier } G \subset \{1..<p\}$
using $\langle \neg \text{prime } p \rangle \text{pa}(2, 3)$ **by** (*auto simp: totatives-def*)

then have $\text{card } (\text{carrier } G) < p - 1$
by (*metis card-atLeastLessThan finite-atLeastLessThan psubset-card-mono*)

show $\text{card } H < (p - 1) \text{ div } 2$
using $\langle \text{card } H \leq \text{card } (\text{carrier } G) \text{ div } 2 \rangle \langle \text{card } (\text{carrier } G) < p - 1 \rangle$


```

    using odd-p less-mult-imp-div-less[of card (carrier G) (p - 1) div 2 2]
    by auto
qed

```

```

lemma prime-imp-G-is-H:
  assumes prime p 2 < p
  shows carrier G = H
  unfolding H-def using assms prime-imp-euler-liar[of p] totatives-less[of - p]
  by (auto simp: totatives-def)

```

end

end

7 Carmichael Numbers

theory Carmichael-Numbers

imports

Residues-Nat

begin

A Carmichael number is a composite number n that Fermat's test incorrectly labels as primes no matter which witness a is chosen (except in the case that a shares a factor with n). [9, 14]

definition *Carmichael-number* :: $nat \Rightarrow bool$ **where**
Carmichael-number $n \iff n > 1 \wedge \neg \text{prime } n \wedge (\forall a. \text{coprime } a \ n \longrightarrow [a \wedge (n - 1) = 1] \pmod n)$

lemma *Carmichael-number-0*[*simp, intro*]: $\neg \text{Carmichael-number } 0$
unfolding *Carmichael-number-def* **by** *simp*

lemma *Carmichael-number-1*[*simp, intro*]: $\neg \text{Carmichael-number } 1$
by (*auto simp: Carmichael-number-def*)

lemma *Carmichael-number-Suc-0*[*simp, intro*]: $\neg \text{Carmichael-number } (\text{Suc } 0)$
by (*auto simp: Carmichael-number-def*)

lemma *Carmichael-number-not-prime*: $\text{Carmichael-number } n \implies \neg \text{prime } n$
by (*auto simp: Carmichael-number-def*)

lemma *Carmichael-number-gt-3*: $\text{Carmichael-number } n \implies n > 3$

proof –

assume *: *Carmichael-number* n

hence $n > 1$ **by** (*auto simp: Carmichael-number-def*)

{

assume $\neg(n > 3)$

with $\langle n > 1 \rangle$ **have** $n = 2 \vee n = 3$ **by** *auto*

with * **and** *Carmichael-number-not-prime*[of n] **have** *False* **by** *auto*

```

}
thus  $n > 3$  by auto
qed

```

The proofs are inspired by [9, 12].

lemma *Carmichael-number-imp-squarefree-aux:*

```

assumes Carmichael-number  $n$ 
assumes  $n: n = p^{\wedge}r * l$  and prime  $p \neg p \text{ dvd } l$ 
assumes  $r > 1$ 
shows False

```

proof –

```

have  $\neg$  prime  $n$  using  $\langle$  Carmichael-number  $n$   $\rangle$  unfolding Carmichael-number-def
by blast

```

```

have * :  $[a^{\wedge}(n-1) = 1] \pmod{n}$  if coprime  $a$   $n$  for  $a$ 
  using  $\langle$  Carmichael-number  $n$   $\rangle$  that
  unfolding Carmichael-number-def
  by blast

```

```

have  $1 \leq n$ 
  unfolding  $n$  using  $\langle$  prime  $p$   $\rangle$   $\langle$   $\neg p \text{ dvd } l$   $\rangle$ 
  by (auto intro: gre1I-nat)

```

```

have  $2 \leq n$ 
proof(cases  $n = 1$ )
  case True
    then show ?thesis
      unfolding  $n$  using  $\langle$   $1 < r$   $\rangle$  prime-gt-1-nat[OF  $\langle$  prime  $p$   $\rangle$ ]
      by simp
  next
    case False
      then show ?thesis using  $\langle$   $1 \leq n$   $\rangle$  by linarith
qed

```

```

have  $p < p^{\wedge}r$ 
  using prime-gt-1-nat[OF  $\langle$  prime  $p$   $\rangle$ ]  $\langle$   $1 < r$   $\rangle$ 
  by (metis power-one-right power-strict-increasing-iff)

```

```

hence  $p < n$  using  $\langle$   $1 \leq n$   $\rangle$  less-le-trans  $n$  by fastforce

```

```

then have [simp]:  $\{..n\} - \{0..Suc\ 0\} = \{2..n\}$  by auto

```

```

obtain  $a$  where  $a: [a = p + 1] \pmod{p^{\wedge}r} [a = 1] \pmod{l}$ 
  using binary-chinese-remainder-nat[of  $p^{\wedge}r$   $l$   $p + 1$   $1$ ]
  and  $\langle$  prime  $p$   $\rangle$  prime-imp-coprime-nat coprime-power-left-iff  $\langle$   $\neg p \text{ dvd } l$   $\rangle$ 
  by blast

```

```

hence coprime  $a$   $n$ 
  using lucas-coprime-lemma[of  $1$   $a$   $l$ ] cong-imp-coprime[of  $p+1$   $a$   $p^{\wedge}r$ ]

```

and *coprime-add-one-left cong-sym*
unfolding $\langle n = p^{\wedge} r * l \rangle$ *coprime-mult-right-iff coprime-power-right-iff power-one-right*
by *blast*

hence $[a^{\wedge} (n - 1) = 1] \text{ (mod } n)$
using $*$ **by** *blast*

hence $[a^{\wedge} (n - 1) = 1] \text{ (mod } p^{\wedge} r)$
using n *cong-modulus-mult-nat* **by** *blast*

hence $A: [a^{\wedge} n = a] \text{ (mod } p^{\wedge} r)$
using *cong-scalar-right*[*of* $a^{\wedge} (n-1)$ 1 $p^{\wedge} r$ a] $\langle 1 \leq n \rangle$
unfolding *power-Suc2*[*symmetric*]
by *simp*

have $r = \text{Suc } (\text{Suc } (r - 2))$
using $\langle 1 < r \rangle$ **by** *linarith*

then have $p^{\wedge} r = p^{\wedge} 2 * p^{\wedge} (r-2)$
by (*simp add: algebra-simps flip: power-add power-Suc*)

hence $[a^{\wedge} n = a] \text{ (mod } p^{\wedge} 2) [a = p + 1] \text{ (mod } p^{\wedge} 2)$
using $\langle 1 < r \rangle$ A *cong-modulus-mult-nat* $\langle [a = p + 1] \text{ (mod } p^{\wedge} r) \rangle$
by *algebra+*

hence $1: [(p + 1)^{\wedge} n = (p + 1)] \text{ (mod } p^{\wedge} 2)$
by (*metis (mono-tags) cong-def power-mod*)

have $[(p + 1)^{\wedge} n = (\sum_{k \leq n} \text{of-nat } (n \text{ choose } k) * p^{\wedge} k * 1^{\wedge} (n - k))] \text{ (mod } p^{\wedge} 2)$
using *binomial*[*of* p 1 n] **by** *simp*

also have $(\sum_{k \leq n} \text{of-nat } (n \text{ choose } k) * p^{\wedge} k * 1^{\wedge} (n - k)) =$
 $(\sum_{k = 0..1} (n \text{ choose } k) * p^{\wedge} k) + (\sum_{k \in \{2..n\}} \text{of-nat } (n \text{ choose } k) * p^{\wedge} k * 1^{\wedge} (n - k))$
using $\langle 2 \leq n \rangle$ *finite-atMost*[*of* n]
by (*subst sum.subset-diff*[**where** $B = \{0..1\}$]) *auto*

also have $[(\sum_{k = 0..1} (n \text{ choose } k) * p^{\wedge} k) = 1] \text{ (mod } p^{\wedge} 2)$
by (*simp add: cong-altdef-nat* $\langle p^{\wedge} r = p^2 * p^{\wedge} (r - 2) \rangle$ n)

also have $[(\sum_{k \in \{2..n\}} \text{of-nat } (n \text{ choose } k) * p^{\wedge} k * 1^{\wedge} (n - k)) = 0] \text{ (mod } p^{\wedge} 2)$
by (*rule cong-eq-0-I*) (*clarsimp simp: cong-0-iff le-imp-power-dvd*)

finally have $2: [(p + 1)^{\wedge} n = 1] \text{ (mod } p^{\wedge} 2)$ **by** *simp*

from *cong-trans*[*OF* *cong-sym*[*OF* 1] 2]
show *?thesis*

```

    using prime-gt-1-nat[OF ‹prime p›]
    by (auto dest: residue-one-dvd[unfolded One-nat-def] simp add: cong-def numeral-2-eq-2)
qed

```

theorem *Carmichael-number-imp-squarefree*:

```

    assumes Carmichael-number n
    shows squarefree n
proof(rule squarefreeI, rule ccontr)
    fix x :: nat
    assume x2 dvd n
    from assms have n > 0 using Carmichael-number-gt-3[of n] by auto
    from ‹x2 dvd n› and ‹0 < n› have 0 < x by auto

```

```

    assume ¬ is-unit x
    then obtain p where prime p p dvd x
      using prime-divisor-exists[of x] ‹0 < x›
      by blast

```

```

    with ‹x2 dvd n› have p2 dvd n
      by auto

```

```

    obtain l where n: n = p ^ multiplicity p n * l
      using multiplicity-dvd[of p n] by blast

```

```

    then have ¬ p dvd l
      using multiplicity-decompose'[where x = n and p = p]
      using ‹prime p› ‹0 < n›
      by (metis nat-dvd-1-iff-1 nat-mult-eq-cancel1 neq0-conv prime-prime-factor-sqrt zero-less-power)

```

```

    have 2 ≤ multiplicity p n
      using ‹p2 dvd n› ‹0 < n› prime-gt-1-nat[OF ‹prime p›]
      by (auto intro!: multiplicity-geI simp: power2-eq-square)

```

```

    then show False
      using Carmichael-number-imp-squarefree-aux[OF ‹Carmichael-number n› n]
      ‹prime p› ‹¬ p dvd l›
      by auto
qed

```

corollary *Carmichael-not-primelow*:

```

    assumes Carmichael-number n
    shows ¬primelow n
    using Carmichael-number-imp-squarefree[of n] Carmichael-number-not-prime[of n] assms
      primelow-gt-0-nat[of n] by (auto simp: not-squarefree-primelow)

```

lemma *Carmichael-number-imp-squarefree-alt-weak*:

assumes *Carmichael-number* n
shows $\exists p l. (n = p * l) \wedge \text{prime } p \wedge \neg p \text{ dvd } l$
proof –
from *assms* **have** $n > 1$
using *Carmichael-number-gt-3*[of n] **by** *simp*
have *squarefree* n
using *Carmichael-number-imp-squarefree* *assms*
by *blast*

obtain $p l$ **where** $p * l = n$ *prime* p $1 < p$
using *assms* *prime-divisor-exists-strong-nat* *prime-gt-1-nat*
unfolding *Carmichael-number-def* **by** *blast*

then have *multiplicity* p $n = 1$
using $\langle 1 < n \rangle$ *squarefree* n **and** *multiplicity-eq-zero-iff*[of n p] *squarefree-factorial-semiring*"[of
 n]
by *auto*

then have $\neg p \text{ dvd } l$
using $\langle 1 < n \rangle$ $\langle \text{prime } p \rangle$ $\langle p * l = n \rangle$ *multiplicity-decompose*"[of n p]
by *force*

show *?thesis*
using $\langle p * l = n \rangle$ $\langle \text{prime } p \rangle$ $\langle \neg p \text{ dvd } l \rangle$
by *blast*
qed

theorem *Carmichael-number-odd*:
assumes *Carmichael-number* n
shows *odd* n
proof (*rule ccontr*)
assume $\neg \text{odd } n$
hence *even* n **by** *simp*
from *assms* **have** $n \geq 4$ **using** *Carmichael-number-gt-3*[of n] **by** *simp*
have $[(n - 1) ^ (n - 1) = n - 1] \pmod n$
using $\langle \text{even } n \rangle$ **and** $\langle n \geq 4 \rangle$ **by** (*intro odd-pow-cong*) *auto*

then have $[(n - 1) ^ (n - 1) \neq 1] \pmod n$
using *cong-trans*[of 1 $(n - 1) ^ (n - 1) n n-1$, *OF cong-sym*] $\langle 4 \leq n \rangle$
by (*auto simp: cong-def*)

moreover have *coprime* $(n - 1) n$
using $\langle n \geq 4 \rangle$ *coprime-diff-one-left-nat*[of n] **by** *auto*

ultimately show *False*
using *assms* **unfolding** *Carmichael-number-def* **by** *blast*
qed

lemma *Carmichael-number-imp-squarefree-alt*:

```

assumes Carmichael-number  $n$ 
shows  $\exists p l. (n = p * l) \wedge \text{prime } p \wedge \neg p \text{ dvd } l \wedge 2 < l$ 
proof –
  obtain  $p l$  where [simp]:  $(n = p * l)$  and  $\text{prime } p \wedge \neg p \text{ dvd } l$ 
    using Carmichael-number-imp-squarefree-alt-weak and assms by blast

  moreover have  $\text{odd } n$  using Carmichael-number-odd and assms by blast

  consider  $l = 0 \vee l = 2 \mid l = 1 \mid 2 < l$ 
    by fastforce

  then have  $2 < l$ 
  proof cases
    case 1
      then show ?thesis
        using  $\langle \text{odd } n \rangle$  by auto
    next
      case 2
        then show ?thesis
          using  $\langle n = p * l \rangle \langle \text{prime } p \rangle \langle \text{Carmichael-number } n \rangle$ 
          unfolding Carmichael-number-def by simp
    qed simp

  ultimately show ?thesis by blast
qed

lemma Carmichael-number-imp-dvd:
  fixes  $n :: \text{nat}$ 
  assumes Carmichael-number: Carmichael-number  $n$  and  $\text{prime } p \wedge p \text{ dvd } n$ 
  shows  $p - 1 \text{ dvd } n - 1$ 
proof –
  have  $\neg \text{prime } n$  using Carmichael-number unfolding Carmichael-number-def
by blast
  obtain  $u$  where  $n = p * u$  using  $\langle p \text{ dvd } n \rangle$  by blast
  have squarefree  $n$  using Carmichael-number-imp-squarefree assms by blast
  then have  $\neg p \text{ dvd } u$ 
    using  $\langle \text{prime } p \rangle \text{not-prime-unit[of } p \rangle$ 
    unfolding power2-eq-square squarefree-def  $\langle n = p * u \rangle$ 
    by fastforce

  define  $R$  where  $R = \text{Residues-nat } p$ 
  interpret residues-nat-prime  $p R$ 
    by unfold-locales (simp-all only:  $\langle \text{prime } p \rangle R\text{-def}$ )

  obtain  $a$  where  $a: a \in \{0 < .. < p\} \text{units.ord } a = p - 1$ 
    using residues-prime-cyclic'  $\langle \text{prime } p \rangle$  by metis
  from  $a$  have  $a \in \text{totatives } p$  by (auto simp: totatives-prime  $\langle \text{prime } p \rangle$ )

  have  $\text{coprime } p u$ 

```

```

using ⟨prime p⟩ ⟨¬ p dvd u⟩
by (simp add: prime-imp-coprime-nat)

then obtain x where [x = a] (mod p) [x = 1] (mod u)
using binary-chinese-remainder-nat[of p u a 1] by blast

have coprime x p
using ⟨a ∈ totatives p⟩ and cong-imp-coprime[OF cong-sym[OF ⟨[x = a] (mod p)⟩]]
by (simp add: coprime-commute totatives-def)
moreover have coprime x u
using coprime-1-left and cong-imp-coprime[OF cong-sym[OF ⟨[x = 1] (mod u)⟩]] by blast
ultimately have coprime x n
by (simp add: ⟨n = p * u⟩)

have [a ^ (n - 1) = x ^ (n - 1)] (mod p)
using ⟨[x = a] (mod p)⟩ by (intro cong-pow) (auto simp: cong-sym-eq)
also have [x ^ (n - 1) = 1] (mod n)
using Carmichael-number ⟨coprime x n⟩ unfolding Carmichael-number-def by
blast
then have [x ^ (n - 1) = 1] (mod p)
using ⟨n = p * u⟩ cong-modulus-mult-nat by blast
finally have ord p a dvd n - 1
by (simp add: ord-divides [symmetric])
also have ord p a = p - 1
using a ⟨a ∈ totatives p⟩ by (simp add: units.ord-residue-mult-group)
finally show ?thesis .
qed

```

The following lemma is also called Korselt's criterion.

```

lemma Carmichael-numberI:
fixes n :: nat
assumes ¬ prime n squarefree n 1 < n and
DIV:  $\bigwedge p. p \in \text{prime-factors } n \implies p - 1 \text{ dvd } n - 1$ 
shows Carmichael-number n
unfolding Carmichael-number-def
proof (intro assms conjI allI impI)
fix a :: nat assume coprime a n

have n: n =  $\prod$  (prime-factors n)
using prime-factorization-nat and squarefree-factorial-semiring'[of n] ⟨1 < n⟩
⟨squarefree n⟩
by fastforce

have x ∈# prime-factorization n  $\implies$  y ∈# prime-factorization n  $\implies$  x ≠ y  $\implies$ 
coprime x y for x y
using in-prime-factors-imp-prime primes-coprime
by blast

```

```

moreover {
  fix  $p$ 
  assume  $p: p \in \# \text{ prime-factorization } n$ 

  have  $\neg p \text{ dvd } a$ 
    using  $\langle \text{coprime } a \ n \rangle p \text{ coprime-common-divisor-nat[of } a \ n \ p]$ 
    by  $(\text{auto simp: in-prime-factors-iff})$ 
  with  $p$  have  $[a \wedge^{(p-1)} = 1] \pmod{p}$ 
    by  $(\text{intro fermat-theorem}) \text{ auto}$ 
  hence  $\text{ord } p \ a \text{ dvd } p - 1$ 
    by  $(\text{subst } (asm) \text{ ord-divides})$ 
  also from  $p$  have  $p - 1 \text{ dvd } n - 1$ 
    by  $(\text{rule DIV})$ 
  finally have  $[a \wedge^{(n-1)} = 1] \pmod{p}$ 
    by  $(\text{subst ord-divides})$ 
}

ultimately show  $[a \wedge^{(n-1)} = 1] \pmod{n}$ 
using  $n \text{ coprime-cong-prod-nat}$  by  $\text{metis}$ 
qed

theorem Carmichael-number-iff:
   $\text{Carmichael-number } n \iff$ 
   $n \neq 1 \wedge \neg \text{prime } n \wedge \text{squarefree } n \wedge (\forall p \in \text{prime-factors } n. p - 1 \text{ dvd } n - 1)$ 
proof –
  consider  $n = 0 \mid n = 1 \mid n > 1$  by  $\text{force}$ 
  thus  $?thesis$  using  $\text{Carmichael-numberI[of } n]$   $\text{Carmichael-number-imp-dvd[of } n]$ 
  by  $\text{cases } (\text{auto simp: Carmichael-number-not-prime Carmichael-number-imp-squarefree})$ 
qed

Every Carmichael number has at least three distinct prime factors.

theorem Carmichael-number-card-prime-factors:
  assumes  $\text{Carmichael-number } n$ 
  shows  $\text{card } (\text{prime-factors } n) \geq 3$ 
proof  $(\text{rule ccontr})$ 
  from  $\text{assms}$  have  $n > 3$ 
    using  $\text{Carmichael-number-gt-3[of } n]$  by  $\text{simp}$ 
  assume  $\neg(\text{card } (\text{prime-factors } n) \geq 3)$ 
  moreover have  $\text{card } (\text{prime-factors } n) \neq 0$ 
    using  $\text{assms Carmichael-number-gt-3[of } n]$  by  $(\text{auto simp: prime-factorization-empty-iff})$ 
  moreover have  $\text{card } (\text{prime-factors } n) \neq 1$ 
    using  $\text{assms}$  by  $(\text{auto simp: one-prime-factor-iff-primelow Carmichael-not-primelow})$ 
  ultimately have  $\text{card } (\text{prime-factors } n) = 2$ 
    by  $\text{linarith}$ 
  then obtain  $p \ q$  where  $pq: \text{prime-factors } n = \{p, q\} \ p \neq q$ 
    by  $(\text{auto simp: card-Suc-eq numeral-2-eq-2})$ 
  hence  $\text{prime } p \ \text{prime } q$  by  $(\text{auto simp: in-prime-factors-iff})$ 

```



```

have n =  $\prod$  (prime-factors n)
  using assms by (subst squarefree-imp-prod-prime-factors-eq)
                    (auto simp: Carmichael-number-imp-squarefree)
with pq have n-eq: n = p * q by simp

have p - 1 dvd n - 1 and q - 1 dvd n - 1 using assms pq
  unfolding Carmichael-number-iff by blast+
with ⟨prime p⟩ ⟨prime q⟩ ⟨n = p * q⟩ ⟨p ≠ q⟩ show False
proof (induction p q rule: linorder-wlog)
  case (le p q)
  hence p < q by auto
  have [q = 1] (mod q - 1)
    using prime-gt-1-nat[of q] ⟨prime q⟩ by (simp add: cong-def le-mod-geq)
  hence [p * q - 1 = p * 1 - 1] (mod q - 1)
    using le prime-gt-1-nat[of p] by (intro cong-diff-nat cong-mult) auto
  hence [p - 1 = n - 1] (mod q - 1)
    by (simp add: ⟨n = p * q⟩ cong-sym-eq)
  also have [n - 1 = 0] (mod q - 1)
    using le by (simp add: cong-def)
  finally have (p - 1) mod (q - 1) = 0
    by (simp add: cong-def)
  also have (p - 1) mod (q - 1) = p - 1
    using prime-gt-1-nat[of p] ⟨prime p⟩ ⟨p < q⟩ by (intro mod-less) auto
  finally show False
    using prime-gt-1-nat[of p] ⟨prime p⟩ by simp
qed (simp-all add: mult.commute)
qed

lemma Carmichael-number-iff':
  fixes n :: nat
  defines P ≡ prime-factorization n
  shows Carmichael-number n  $\longleftrightarrow$ 
        n > 1  $\wedge$  size P ≠ 1  $\wedge$  ( $\forall p \in \#P. \text{count } P \text{ } p = 1 \wedge p - 1 \text{ } \text{dvd } n - 1$ )
  unfolding Carmichael-number-iff
  by (cases n = 0) (auto simp: P-def squarefree-factorial-semiring' count-prime-factorization)

The smallest Carmichael number is 561, and it was found and proven so by
Carmichael in 1910 [6].

lemma Carmichael-number-561: Carmichael-number 561 (is Carmichael-number
?n)
proof -
  have [simp]: prime-factorization (561 :: nat) = {#3, 11, 17#}
    by (rule prime-factorization-eqI) auto
  show ?thesis by (subst Carmichael-number-iff') auto
qed

end

```

8 Fermat Witnesses

theory *Fermat-Witness*

imports *Euler-Witness Carmichael-Numbers*

begin

definition *divide-out* :: 'a :: factorial-semiring \Rightarrow 'a \Rightarrow 'a \times nat **where**
divide-out p x = (x div p \wedge multiplicity p x, multiplicity p x)

lemma *fst-divide-out* [*simp*]: *fst* (divide-out p x) = x div p \wedge multiplicity p x
and *snd-divide-out* [*simp*]: *snd* (divide-out p x) = multiplicity p x
by (*simp-all add: divide-out-def*)

function *divide-out-aux* :: 'a :: factorial-semiring \Rightarrow 'a \times nat \Rightarrow 'a \times nat **where**
divide-out-aux p (x, acc) =
 (if x = 0 \vee is-unit p \vee \neg p dvd x then (x, acc) else divide-out-aux p (x div p,
 acc + 1))

by *auto*

termination proof (relation measure (λ (p, x, -). multiplicity p x))

fix p x :: 'a **and** acc :: nat

assume \neg (x = 0 \vee is-unit p \vee \neg p dvd x)

thus ((p, x div p, acc + 1), p, x, acc) \in measure (λ (p, x, -). multiplicity p x)

by (*auto elim!: dvdE simp: multiplicity-times-same*)

qed *auto*

lemmas [*simp del*] = *divide-out-aux.simps*

lemma *divide-out-aux-correct*:

divide-out-aux p z = (fst z div p \wedge multiplicity p (fst z), snd z + multiplicity p
 (fst z))

proof (*induction* p z rule: *divide-out-aux.induct*)

case (1 p x acc)

show ?case

proof (*cases* x = 0 \vee is-unit p \vee \neg p dvd x)

case *False*

have x div p div p \wedge multiplicity p (x div p) = x div p \wedge multiplicity p x

using *False*

by (*subst dvd-div-mult2-eq [symmetric]*)

(*auto elim!: dvdE simp: multiplicity-dvd multiplicity-times-same*)

with *False* **show** ?thesis **using** 1

by (*subst divide-out-aux.simps*)

(*auto elim: dvdE simp: multiplicity-times-same multiplicity-unit-left
 not-dvd-imp-multiplicity-0*)

qed (*auto simp: divide-out-aux.simps multiplicity-unit-left not-dvd-imp-multiplicity-0*)

qed

lemma *divide-out-code* [*code*]: *divide-out* p x = *divide-out-aux* p (x, 0)

by (*simp add: divide-out-aux-correct divide-out-def*)

lemma *multiplicity-code* [code]: $\text{multiplicity } p \ x = \text{snd } (\text{divide-out-aux } p \ (x, 0))$
by (*simp add: divide-out-aux-correct*)

lemma *multiplicity-times-same-power*:
assumes $x \neq 0 \ \neg \text{is-unit } p \ p \neq 0$
shows $\text{multiplicity } p \ (p^k * x) = \text{multiplicity } p \ x + k$
using *assms* **by** (*induction k*) (*simp-all add: multiplicity-times-same mult.assoc*)

lemma *divide-out-unique-nat*:
fixes $n :: \text{nat}$
assumes $\neg \text{is-unit } p \ p \neq 0 \ \neg p \ \text{dvd } m$ **and** $n = p^k * m$
shows $m = n \ \text{div } p^k$ **and** $k = \text{multiplicity } p \ n$
proof –
from *assms* **have** $n \neq 0$ **by** (*intro notI*) *auto*
thus $*$: $k = \text{multiplicity } p \ n$
using *assms* **by** (*auto simp: assms multiplicity-times-same-power not-dvd-imp-multiplicity-0*)
from *assms* **show** $m = n \ \text{div } p^k$
unfolding $*$ [*symmetric*] **by** *auto*
qed

context
fixes $a \ n :: \text{nat}$
begin

definition *fermat-liar* $\longleftrightarrow [a^{n-1} = 1] \ (\text{mod } n)$
definition *fermat-witness* $\longleftrightarrow [a^{n-1} \neq 1] \ (\text{mod } n)$

definition *strong-fermat-liar* \longleftrightarrow
 $(\forall k \ m. \ \text{odd } m \longrightarrow n - 1 = 2^k * m \longrightarrow$
 $[a^m = 1] \ (\text{mod } n) \vee (\exists i \in \{0..< k\}. [a^{2^i * m} = n-1] \ (\text{mod } n)))$

definition *strong-fermat-witness* $\longleftrightarrow \neg \text{strong-fermat-liar}$

lemma *fermat-liar-witness-of-composition*[*iff*]:
 $\neg \text{fermat-liar} = \text{fermat-witness}$
 $\neg \text{fermat-witness} = \text{fermat-liar}$
unfolding *fermat-liar-def* **and** *fermat-witness-def*
by *simp-all*

lemma *strong-fermat-liar-code* [code]:
 $\text{strong-fermat-liar} \longleftrightarrow$
 $(\text{let } (m, k) = \text{divide-out } 2 \ (n - 1)$
 $\text{in } [a^m = 1] \ (\text{mod } n) \vee (\exists i \in \{0..< k\}. [a^{2^i * m} = n-1] \ (\text{mod } n)))$
(is ?lhs = ?rhs)
proof (*cases n > 1*)

```

case True
define m where  $m = (n - 1) \text{ div } 2 \wedge \text{multiplicity } 2 (n - 1)$ 
define k where  $k = \text{multiplicity } 2 (n - 1)$ 
have mk:  $\text{odd } m \wedge n - 1 = 2 \wedge k * m$ 
  using True multiplicity-decompose[of  $n - 1$  2] multiplicity-dvd[of 2  $n - 1$ ]
  by (auto simp: m-def k-def)
show ?thesis
proof
  assume ?lhs
  hence  $[a \wedge^m = 1] \pmod{n} \vee (\exists i \in \{0..<k\}. [a \wedge^{2 \wedge i * m} = n-1] \pmod{n})$ 
    using True mk by (auto simp: divide-out-def strong-fermat-liar-def)
  thus ?rhs by (auto simp: Let-def divide-out-def m-def k-def)
next
  assume ?rhs
  thus ?lhs using divide-out-unique-nat[of 2]
    by (auto simp: strong-fermat-liar-def divide-out-def)
qed
qed (auto simp: strong-fermat-liar-def divide-out-def)

```

```

context
  assumes  $*$  :  $a \in \{1 ..<n\}$ 
begin

```

```

lemma strong-fermat-witness-iff:
  strong-fermat-witness =
     $(\exists k m. \text{odd } m \wedge n - 1 = 2 \wedge k * m \wedge [a \wedge^m \neq 1] \pmod{n} \wedge$ 
       $(\forall i \in \{0..<k\}. [a \wedge^{2 \wedge i * m} \neq n-1] \pmod{n}))$ 
  unfolding strong-fermat-witness-def strong-fermat-liar-def
  by blast

```

```

lemma not-coprime-imp-witness:  $\neg \text{coprime } a \ n \implies \text{fermat-witness}$ 
  using  $*$  lucas-coprime-lemma[of  $n-1$   $a$   $n$ ]
  by (auto simp: fermat-witness-def)

```

```

corollary liar-imp-coprime: fermat-liar  $\implies$  coprime  $a \ n$ 
  using not-coprime-imp-witness fermat-liar-witness-of-composition by blast

```

```

lemma fermat-witness-imp-strong-fermat-witness:

```

```

  assumes  $1 < n$  fermat-witness
  shows strong-fermat-witness
proof –
  from  $\langle \text{fermat-witness} \rangle$  have  $[a \wedge^{(n-1)} \neq 1] \pmod{n}$ 
    unfolding fermat-witness-def .

```

```

  obtain k m where  $\text{odd } m$  and  $n: n - 1 = 2 \wedge k * m$ 
    using  $*$  by (auto intro: multiplicity-decompose'[of  $(n-1)$  2])

```

```

  moreover have  $[a \wedge^m \neq 1] \pmod{n}$ 

```

```

using ⟨[a^(n-1) ≠ 1] (mod n)⟩ n ord-divides by auto

moreover {
  fix i :: nat
  assume i ∈ {0..<k}
  hence i ≤ k - 1 0 < k by auto
  then have [a^(2^i * m) ≠ n - 1] (mod n) [a^(2^i * m) ≠ 1] (mod n)
  proof(induction i rule: inc-induct)
    case base
      have * : a^(n - 1) = (a^(2^(k - 1) * m))^2
      using ⟨0 < k⟩ n semiring-normalization-rules(27)[of 2 :: nat k - 1]
      by (auto simp flip: power-even-eq simp add: algebra-simps)

      case 1
      from * show ?case
      using ⟨[a^(n-1) ≠ 1] (mod n)⟩ and square-minus-one-cong-one[OF ⟨1 <
n⟩] by auto

      case 2
      from * show ?case using n ⟨[a^(n-1) ≠ 1] (mod n)⟩ and square-one-cong-one
by metis
    next
      case (step q)
      then have
        IH2: [a^(2^Suc q * m) ≠ 1] (mod n) using ⟨0 < k⟩ by blast+

      have * : a^(2^Suc q * m) = (a^(2^q * m))^2
      by (auto simp flip: power-even-eq simp add: algebra-simps)

      case 1
      from * show ?case using IH2 and square-minus-one-cong-one[OF ⟨1 < n⟩]
by auto
      case 2
      from * show ?case using IH2 and square-one-cong-one by metis
    qed
  }

ultimately show ?thesis unfolding strong-fermat-witness-iff by blast
qed

corollary strong-fermat-liar-imp-fermat-liar:
  assumes 1 < n strong-fermat-liar
  shows fermat-liar
  using fermat-witness-imp-strong-fermat-witness assms
  and fermat-liar-witness-of-composition strong-fermat-witness-def
  by blast

lemma euler-liar-is-fermat-liar:

```

assumes $1 < n$ *euler-liar* a n *coprime* a n *odd* n
shows *fermat-liar*
proof –
have $[Jacobi\ a\ n = a^{((n - 1)\ div\ 2)}] \pmod n$
using $\langle euler-liar\ a\ n \rangle$ **unfolding** *euler-witness-def* **by** *simp*

hence $[(Jacobi\ a\ n)^2 = (a^{((n - 1)\ div\ 2)})^2] \pmod n$
by (*simp add: cong-pow*)

moreover have $Jacobi\ a\ n \in \{1, -1\}$
using *Jacobi-values* *Jacobi-eq-0-iff-not-coprime*[*of* n] $\langle coprime\ a\ n \rangle$ $\langle 1 < n \rangle$
by *force*

ultimately have $[1 = (a^{((n - 1)\ div\ 2)})^2] \pmod n$
using *cong-int-iff* **by** *force*

also have $(a^{((n - 1)\ div\ 2)})^2 = a^{(n - 1)}$
using $\langle odd\ n \rangle$ **by** (*simp flip: power-mult*)

finally show *?thesis*
using *cong-sym* *fermat-liar-def*
by *blast*
qed

corollary *fermat-witness-is-euler-witness*:
assumes $1 < n$ *fermat-witness* *coprime* a n *odd* n
shows *euler-witness* a n
using *assms* *euler-liar-is-fermat-liar* *fermat-liar-witness-of-composition*
by *blast*

end
end

lemma *one-is-fermat-liar*[*simp*]: $1 < n \implies$ *fermat-liar* 1 n
using *fermat-liar-def* **by** *auto*

lemma *one-is-strong-fermat-liar*[*simp*]: $1 < n \implies$ *strong-fermat-liar* 1 n
using *strong-fermat-liar-def* **by** *auto*

lemma *prime-imp-fermat-liar*:
 $prime\ p \implies a \in \{1 ..< p\} \implies$ *fermat-liar* a p
unfolding *fermat-liar-def*
using *fermat-theorem* **and** *nat-dvd-not-less* **by** *simp*

lemma *not-Carmichael-numberD*:
assumes \neg *Carmichael-number* n \neg *prime* n **and** $1 < n$
shows $\exists a \in \{2 ..< n\} .$ *fermat-witness* a $n \wedge$ *coprime* a n
proof –
obtain a **where** *coprime* a n $[a^{(n-1)} \neq 1] \pmod n$

using *assms* **unfolding** *Carmichael-number-def* **by** *blast*

moreover from *this* **and** $\langle 1 < n \rangle$ **have** $a \bmod n \in \{1..<n\}$
by (*cases* $a = 0$) (*auto intro!* : *gre1I-nat*)

ultimately have $a \bmod n \in \{1 ..< n\}$ *coprime* $(a \bmod n) n [(a \bmod n) \wedge^{(n-1)} \neq 1] \pmod n$
using $\langle 1 < n \rangle$ **by** *simp-all*

hence *fermat-witness* $(a \bmod n) n$
using *fermat-witness-def* **by** *blast*

hence $1 \neq (a \bmod n)$
using $\langle 1 < n \rangle$ $\langle (a \bmod n) \in \{1 ..< n\} \rangle$ **and** *one-is-fermat-liar* *fermat-liar-witness-of-composition*(1)
by *metis*

thus *?thesis*
using \langle *fermat-witness* $(a \bmod n) n \rangle$ \langle *coprime* $(a \bmod n) n \rangle$ $\langle (a \bmod n) \in \{1..<n\} \rangle$
by *fastforce*

qed

proposition *prime-imp-strong-fermat-witness*:
fixes $p :: \text{nat}$
assumes *prime* p $2 < p$ $a \in \{1 ..< p\}$
shows *strong-fermat-liar* a p

proof –

{ **fix** $k m :: \text{nat}$
define j **where** $j \equiv \text{LEAST } k. [a \wedge^{(2^k * m)} = 1] \pmod p$

have $[a \wedge^{(p-1)} = 1] \pmod p$
using *fermat-theorem*[*OF* \langle *prime* $p \rangle$, *of* a] $\langle a \in \{1 ..< p\} \rangle$ **by** *force*

moreover assume *odd* m **and** $n: p - 1 = 2^k * m$

ultimately have $[a \wedge^{(2^k * m)} = 1] \pmod p$ **by** *simp*

moreover assume $[a \wedge^m \neq 1] \pmod p$
then have $0 < x$ **if** $[a \wedge^{(2^x * m)} = 1] \pmod p$ **for** x
using *that* **by** (*auto intro:* *gr0I*)

ultimately have $0 < j$ $j \leq k$ $[a \wedge^{(2^j * m)} = 1] \pmod p$
using *LeastI2*[*of* - k ($<$) 0] *Least-le*[*of* - k] *LeastI*[*of* - k]
by (*simp-all* *add:* *j-def*)

moreover from *this* **have** $[a \wedge^{(2^{(j-1)} * m)} \neq 1] \pmod p$
using *not-less-Least*[*of* $j - 1$ $\lambda k. [a \wedge^{(2^k * m)} = 1] \pmod p$]
unfolding *j-def* **by** *simp*

moreover have $a^{(2^{(j-1)} * m)} * a^{(2^{(j-1)} * m)} = a^{(2^j * m)}$
using $\langle 0 < j \rangle$
by (*simp add: algebra-simps semiring-normalization-rules(27) flip: power2-eq-square power-even-eq*)

ultimately have $(j-1) \in \{0..<k\}$ $[a^{(2^{(j-1)} * m)} = p - 1] \pmod{p}$
using *cong-square-alt[OF $\langle \text{prime } p \rangle$, of $a^{(2^{(j-1)} * m)}$]*
by *simp-all*
}

then show *?thesis unfolding strong-fermat-liar-def by blast*
qed

lemma *ignore-one:*

fixes $P :: - \Rightarrow \text{nat} \Rightarrow \text{bool}$
assumes $P \ 1 \ n \ 1 < n$
shows $\text{card } \{a \in \{1..<n\}. P \ a \ n\} = 1 + \text{card } \{a. 2 \leq a \wedge a < n \wedge P \ a \ n\}$
proof –
have $\text{insert } 1 \ \{a. 2 \leq a \wedge a < n \wedge P \ a \ n\} = \{a. 1 \leq a \wedge a < n \wedge P \ a \ n\}$
using *assms by auto*

moreover have $\text{card } (\text{insert } 1 \ \{a. 2 \leq a \wedge a < n \wedge P \ a \ n\}) = \text{Suc } (\text{card } \{a. 2 \leq a \wedge a < n \wedge P \ a \ n\})$
using *card-insert-disjoint by auto*

ultimately show *?thesis by force*
qed

Proofs in the following section are inspired by [10, 7, 1].

proposition *not-Carmichael-number-imp-card-fermat-witness-bound:*

fixes $n :: \text{nat}$
assumes $\neg \text{prime } n \ \neg \text{Carmichael-number } n \ \text{odd } n \ 1 < n$
shows $(n-1) \ \text{div } 2 < \text{card } \{a \in \{1 ..<n\}. \text{fermat-witness } a \ n\}$
and $(\text{card } \{a. 2 \leq a \wedge a < n \wedge \text{strong-fermat-liar } a \ n\}) < \text{real } (n - 2) / 2$
and $(\text{card } \{a. 2 \leq a \wedge a < n \wedge \text{fermat-liar } a \ n\}) < \text{real } (n - 2) / 2$
proof –

define G **where** $G = \text{Residues-Mult } n$
interpret *residues-mult-nat* $n \ G$
by *unfold-locales (use $\langle n > 1 \rangle$ in $\langle \text{simp-all only: } G\text{-def} \rangle$)*
define h **where** $h \equiv \lambda a. a^{(n-1)} \pmod{n}$
define ker **where** $\text{ker} = \text{kernel } G \ (G(\text{carrier} := h \ \text{carrier } G)) \ h$

have *finite ker by (auto simp: ker-def kernel-def)*
moreover have $1 \in \text{ker}$ **using** $\langle n > 1 \rangle$ **by** *(auto simp: ker-def kernel-def h-def)*
ultimately have *[simp]: card ker > 0*
by *(subst card-gt-0-iff) auto*

have *totatives-eq: totatives n = {k ∈ {1..<n}. coprime k n}*


```

    using totatives-less[of - n] ⟨n > 1⟩ by (force simp: totatives-def)
  have ker-altdef: ker = {a ∈ {1..<n}. fermat-liar a n}
  unfolding ker-def fermat-liar-def carrier-eq kernel-def totatives-eq using ⟨n >
1⟩
  by (force simp: h-def cong-def intro: coprimeI-power-mod)

  have h-is-hom: h ∈ hom G G
  unfolding hom-def using nat-pow-closed
  by (auto simp: h-def power-mult-distrib mod-simps)
  then interpret h: group-hom G G h
  by unfold-locales

  obtain a where a: a ∈ {2..<n} fermat-witness a n coprime a n
  using assms power-one not-Carmichael-numberD by blast

  have h a ≠ 1 using a by (auto simp: fermat-witness-def cong-def h-def)
  hence 2 ≤ card {1, h a} by simp
  also have ... ≤ card (h ` carrier G)
  proof (intro card-mono; safe?)
    from ⟨n > 1⟩ have 1 = h 1 by (simp add: h-def)
    also have ... ∈ h ` carrier G by (intro imageI) (use ⟨n > 1⟩ in auto)
    finally show 1 ∈ h ` carrier G .
  next
    show h a ∈ h ` carrier G
    using a by (intro imageI) (auto simp: totatives-def)
  qed auto
  also have ... * card ker = order G
  using homomorphism-thm-order[OF h.group-hom-axioms] by (simp add: ker-def
order-def)
  also have order G < n - 1
  using totient-less-not-prime[of n] assms by (simp add: order-eq)
  finally have card ker < (n - 1) div 2
  using ⟨odd n⟩ by (auto elim!: oddE)

  have (n - 1) div 2 < (n - 1) - card ker
  using ⟨card ker < (n - 1) div 2⟩ by linarith
  also have ... = card ({1..<n} - ker)
  by (subst card-Diff-subset) (auto simp: ker-altdef)
  also have {1..<n} - ker = {a ∈ {1..<n}. fermat-witness a n}
  by (auto simp: fermat-witness-def fermat-liar-def ker-altdef)
  finally show (n - 1) div 2 < card {a ∈ {1..<n}. fermat-witness a n} .

  have card {a. 2 ≤ a ∧ a < n ∧ fermat-liar a n} ≤ card (ker - {1})
  by (intro card-mono) (auto simp: ker-altdef fermat-liar-def fermat-witness-def)
  also have ... = card ker - 1
  using ⟨n > 1⟩ by (subst card-Diff-subset) (auto simp: ker-altdef fermat-liar-def)
  also have ... < (n - 2) div 2
  using ⟨card ker < (n - 1) div 2⟩ ⟨odd n⟩ ⟨card ker > 0⟩ by linarith
  finally show *: card {a. 2 ≤ a ∧ a < n ∧ fermat-liar a n} < real (n - 2) / 2

```

by simp
have $\text{card } \{a. 2 \leq a \wedge a < n \wedge \text{strong-fermat-liar } a \ n\} \leq$
 $\text{card } \{a. 2 \leq a \wedge a < n \wedge \text{fermat-liar } a \ n\}$
by (intro card-mono) (auto intro!: strong-fermat-liar-imp-fermat-liar)
moreover note *
ultimately show $\text{card } \{a. 2 \leq a \wedge a < n \wedge \text{strong-fermat-liar } a \ n\} < \text{real } (n$
 $- 2) / 2$
by simp
qed

proposition *Carmichael-number-imp-lower-bound-on-strong-fermat-witness:*

fixes $n :: \text{nat}$
assumes *Carmichael-number: Carmichael-number* n
shows $(n - 1) \text{ div } 2 < \text{card } \{a \in \{1..<n\}. \text{strong-fermat-witness } a \ n\}$
and $\text{real } (\text{card } \{a. 2 \leq a \wedge a < n \wedge \text{strong-fermat-liar } a \ n\}) < \text{real } (n - 2)$
 $/ 2$

proof –

from *assms* **have** $n > 3$ **by** (intro *Carmichael-number-gt-3*)
hence $n - 1 \neq 0$ *¬is-unit* $(2 :: \text{nat})$ **by** *auto*
obtain $k \ m$ **where** *odd* m **and** *n-less: $n - 1 = 2^k * m$*
using *multiplicity-decompose* [OF $\langle n - 1 \neq 0 \rangle$ $\langle \text{¬is-unit } (2 :: \text{nat}) \rangle$] **by** *metis*

obtain $p \ l$ **where** $n = p * l$ **and** *prime* p $\neg p \text{ dvd } l$ $2 < l$
using *Carmichael-number-imp-squarefree-alt* [OF *Carmichael-number*]
by *blast*

then have *coprime* $p \ l$ **using** *prime-imp-coprime-nat* **by** *blast*

have *odd* n **using** *Carmichael-number-odd Carmichael-number* **by** *simp*

have $2 < n$ **using** $\langle n > 3 \rangle$ $\langle \text{odd } n \rangle$ **by** *presburger*

note *prime-gt-1-nat* [OF $\langle \text{prime } p \rangle$]

have $2 < p$ **using** $\langle \text{odd } n \rangle$ n $\langle \text{prime } p \rangle$ *prime-ge-2-nat*
and *dvd-triv-left le-neq-implies-less* **by** *blast*

let $?P = \lambda k. (\forall a. \text{coprime } a \ p \longrightarrow [a^{2^k * m} = 1] \text{ (mod } p))$

define j **where** $j \equiv \text{LEAST } k. ?P \ k$

define H **where** $H \equiv \{a \in \{1..<n\}. \text{coprime } a \ n \wedge ([a^{2^{j-1} * m} = 1]$
 $\text{(mod } n) \vee$

$[a^{2^{j-1} * m} = n - 1] \text{ (mod } n))\}$

have $k : \forall a. \text{coprime } a \ n \longrightarrow [a^{2^k * m} = 1] \text{ (mod } n)$

using *Carmichael-number unfolding Carmichael-number-def n-less* **by** *blast*

obtain $k' m'$ **where** *odd* m' **and** *p-less*: $p - 1 = 2^{\wedge k'} * m'$
using $\langle 1 < p \rangle$ **by** (*auto intro: multiplicity-decompose'*[of $(p-1) 2$])

have $p - 1 \text{ dvd } n - 1$
using *Carmichael-number-imp-dvd*[OF *Carmichael-number* $\langle \text{prime } p \rangle$] $\langle n = p$
 $* l \rangle$
by *fastforce*

then have $p - 1 \text{ dvd } 2^{\wedge k'} * m$
unfolding *n-less p-less*
using $\langle \text{odd } m \rangle$ $\langle \text{odd } m' \rangle$
and *coprime-dvd-mult-left-iff*[of $2^{\wedge k'} m 2^{\wedge k}$] *coprime-dvd-mult-right-iff*[of m'
 $2^{\wedge k} m$]
by *auto*

have k' : $\forall a. \text{coprime } a p \longrightarrow [a^{\wedge (2^{\wedge k'} * m)} = 1] \pmod p$
proof *safe*
fix a
assume *coprime* $a p$
hence $\neg p \text{ dvd } a$ **using** *p-coprime-right-nat*[OF $\langle \text{prime } p \rangle$] **by** *simp*

have $[a^{\wedge (2^{\wedge k'} * m')} = 1] \pmod p$
unfolding *p-less[symmetric]*
using *fermat-theorem* $\langle \text{prime } p \rangle$ $\langle \neg p \text{ dvd } a \rangle$ **by** *blast*

then show $[a^{\wedge (2^{\wedge k'} * m)} = 1] \pmod p$
using $\langle p - 1 \text{ dvd } 2^{\wedge k'} * m \rangle$
unfolding *p-less n-less*
by (*meson dvd-trans ord-divides*)

qed

have *j-prop*: $[a^{\wedge (2^{\wedge j} * m)} = 1] \pmod p$ **if** *coprime* $a p$ **for** a
using *that LeastI*[of $?P k'$, OF k' , *folded j-def*] *cong-modulus-mult coprime-mult-right-iff*

unfolding *j-def n* **by** *blast*

have *j-least*: $[a^{\wedge (2^{\wedge i} * m)} = 1] \pmod p$ **if** *coprime* $a p$ $j \leq i$ **for** a i
proof –
obtain c **where** $i = j + c$ **using** *le-iff-add*[of $j i$] $\langle j \leq i \rangle$ **by** *blast*

then have $[a^{\wedge (2^{\wedge i} * m)} = a^{\wedge (2^{\wedge (j + c)} * m)}] \pmod p$ **by** *simp*

also have $[a^{\wedge (2^{\wedge (j + c)} * m)} = (a^{\wedge (2^{\wedge j} * m)})^{\wedge (2^{\wedge c})}] \pmod p$
by (*simp flip: power-mult add: algebra-simps power-add*)

also note *j-prop*[OF $\langle \text{coprime } a p \rangle$]

also have $[1^{\wedge (2^{\wedge c)} = 1] \pmod p$ **by** *simp*

finally show *?thesis* .
qed

have *neq-p*: $[p - 1 \neq 1](\text{mod } p)$
using $\langle 2 < p \rangle$ **and** *cong-less-modulus-unique-nat*[*of p-1 1 p*]
by *linarith*

have $0 < j$
proof (*rule LeastI2*[*of ?P k', OF k', folded j-def*], *rule grOI*)
fix *x*
assume $\forall a. \text{coprime } a \ p \longrightarrow [a \wedge (2 \wedge x * m) = 1](\text{mod } p)$
then have $[(p - 1) \wedge (2 \wedge x * m) = 1](\text{mod } p)$
using *coprime-diff-one-left-nat*[*of p*] *prime-gt-1-nat*[*OF* $\langle \text{prime } p \rangle$]
by *simp*

moreover assume $x = 0$
hence $[(p-1) \wedge (2 \wedge x * m) = p - 1](\text{mod } p)$
using $\langle \text{odd } m \rangle$ *odd-pow-cong*[*OF* - $\langle \text{odd } m \rangle$, *of p*] *prime-gt-1-nat*[*OF* $\langle \text{prime } p \rangle$]
]
by *auto*

ultimately show *False*
using $\langle [p - 1 \neq 1](\text{mod } p) \rangle$ **by** (*simp add: cong-def*)
qed

then have $j - 1 < j$ **by** *simp*

then obtain *x* **where** *coprime x p* $[x \wedge (2 \wedge (j-1) * m) \neq 1](\text{mod } p)$
using *not-less-Least*[*of j - 1 ?P, folded j-def*] **unfolding** *j-def* **by** *blast*

define *G* **where** $G = \text{Residues-Mult } n$
interpret *residues-mult-nat n G*
by *unfold-locales* (*use* $\langle n > 3 \rangle$ **in** $\langle \text{simp-all only: } G\text{-def} \rangle$)

have *H-subset*: $H \subseteq \text{carrier } G$ **unfolding** *H-def* **by** (*auto simp: totatives-def*)

from $\langle n > 3 \rangle$ **have** $\langle n > 1 \rangle$ **by** *simp*
interpret *H*: *subgroup H G*
proof (*rule subgroupI, goal-cases*)
case 1
then show *?case* **using** *H-subset* .
next
case 2
then show *?case* **unfolding** *H-def* **using** $\langle 1 < n \rangle$ **by** *force*
next
case (3 *a*)
define *y* **where** $y = \text{inv}_G \ a$

then have $y \in \text{carrier } G$

```

using H-subset ⟨a ∈ H⟩ by (auto simp del: carrier-eq)

then have 1 ≤ y y < n coprime y n
  using totatives-less[of y n] ⟨n > 3⟩ by (auto simp: totatives-def)

moreover have [y ^ (2 ^ (j - Suc 0) * m) = Suc 0] (mod n)
  if [y ^ (2 ^ (j - Suc 0) * m) ≠ n - Suc 0] (mod n)
proof -
  from ⟨a ∈ H⟩ have [a * y = 1] (mod n)
    using H-subset r-inv[of a] y-def by (auto simp: cong-def)
  hence [(a * y) ^ (2 ^ (j - 1) * m) = 1 ^ (2 ^ (j - 1) * m)] (mod n)
    by (intro cong-pow)
  hence [(a * y) ^ (2 ^ (j - 1) * m) = 1] (mod n)
    by simp
  hence *: [a ^ (2 ^ (j - 1) * m) * y ^ (2 ^ (j - 1) * m) = 1] (mod n)
    by (simp add: power-mult-distrib)
  from ⟨a ∈ H⟩ have 1 ≤ a a < n coprime a n
    unfolding H-def by auto

  have [a ^ (2 ^ (j - 1) * m) = 1] (mod n) ∨ [a ^ (2 ^ (j - 1) * m) = n -
1] (mod n)
    using ⟨a ∈ H⟩ by (auto simp: H-def)
  thus ?thesis
  proof
    note *
    also assume [a ^ (2 ^ (j - 1) * m) = 1] (mod n)
    finally show ?thesis by simp
  next
    assume [a ^ (2 ^ (j - 1) * m) = n - 1] (mod n)
    then have [y ^ (2 ^ (j - 1) * m) = n - 1] (mod n)
      using minus-one-cong-solve[OF ⟨1 < n⟩] * ⟨coprime a n⟩ ⟨coprime y n
⟩ coprime-power-left-iff
      by blast+
    thus ?thesis using that by simp
  qed
qed

ultimately show ?case using ⟨a ∈ H⟩ unfolding H-def y-def by auto
next
case (4 a b)
hence a ∈ totatives n b ∈ totatives n
  by (auto simp: H-def totatives-def)
hence a * b mod n ∈ totatives n
  using m-closed[of a b] by simp
hence a * b mod n ∈ {1..<n} coprime (a * b) n
  using totatives-less[of a * b n] ⟨n > 3⟩ by (auto simp: totatives-def)

moreover define x y where x = a ^ (2 ^ (j - 1) * m) and y = b ^ (2 ^ (j
- 1) * m)

```

```

have [x * y = 1] (mod n) ∨ [x * y = n - 1] (mod n)
proof -
  have *: x mod n ∈ {1, n - 1} y mod n ∈ {1, n - 1}
    using 4 by (auto simp: H-def x-def y-def cong-def)
  have [x * y = (x mod n) * (y mod n)] (mod n)
    by (intro cong-mult) auto
  moreover have ((x mod n) * (y mod n)) mod n ∈ {1, n - 1}
    using * square-minus-one-cong-one[OF ‹1 < n› ‹n > 1›] by (auto simp:
cong-def)
  ultimately show ?thesis using ‹n > 1› by (simp add: cong-def mod-simps)
qed

ultimately show ?case by (auto simp: H-def x-def y-def power-mult-distrib)
qed

{ obtain a where [a = x] (mod p) [a = 1] (mod l) a < p * l
  using binary-chinese-remainder-unique-nat[of p l x 1]
  and ‹¬ p dvd l› ‹prime p› prime-imp-coprime-nat
  by auto

  moreover have coprime a p
    using ‹coprime x p› cong-imp-coprime[OF cong-sym[OF ‹[a = x] (mod p)›]]
  coprime-mult-right-iff
  unfolding n by blast

  moreover have coprime a l
    using coprime-1-left cong-imp-coprime[OF cong-sym[OF ‹[a = 1] (mod l)›]]
    by blast

  moreover from ‹prime p› and ‹coprime a p› have a > 0
    by (intro Nat.gr0I) auto

  ultimately have a ∈ carrier G
    using ‹2 < l› by (auto intro: gr1I-nat simp: n_totatives-def)

  have [a ^ (2^(j-1)) * m ≠ 1] (mod p)
    using ‹[x^(2^(j-1)) * m ≠ 1] (mod p)› ‹[a = x] (mod p)› and cong-trans
  cong-pow cong-sym
  by blast

  then have [a ^ (2^(j-1)) * m ≠ 1] (mod n)
    using cong-modulus-mult-nat n by fast

  moreover
  have [a ^ (2^(j - Suc 0)) * m ≠ n - 1] (mod n)
  proof -
    have [a ^ (2^(j - 1)) * m = 1] (mod l)
      using cong-pow[OF ‹[a = 1] (mod l)›] by auto

```

moreover have $Suc\ 0 \neq (n - Suc\ 0) \bmod l$
using $\langle 2 < l \rangle \langle odd\ n \rangle$
by (*metis mod-Suc-eq mod-less mod-mult-self2-is-0 numeral-2-eq-2 odd-Suc-minus-one zero-neq-numeral*)

then have $[1 \neq n - 1] \pmod{l}$
using $\langle 2 < l \rangle \langle odd\ n \rangle$ **unfolding** *cong-def* **by** *simp*

moreover have $l \neq Suc\ 0$ **using** $\langle 2 < l \rangle$ **by** *simp*

ultimately have $[a \wedge (2 \wedge (j - Suc\ 0) * m) \neq n - 1] \pmod{l}$
by (*auto simp add: cong-def n mod-simps dest: cong-modulus-mult-nat*)

then show *?thesis*
using *cong-modulus-mult-nat mult commute n* **by** *metis*
qed

ultimately have $a \notin H$ **unfolding** *H-def* **by** *auto*

hence $H \subset carrier\ (G)$
using *H-subset subgroup.mem-carrier* **and** $\langle a \in carrier\ (G) \rangle$
by *fast*

}

have $card\ H \leq order\ G\ div\ 2$
by (*intro proper-subgroup-imp-bound-on-card*) (*use* $\langle H \subset carrier\ G \rangle$ *H.is-subgroup*
in *auto*)

also from *assms* **have** $\neg prime\ n$ **by** (*auto dest: Carmichael-number-not-prime*)

hence $order\ G\ div\ 2 < (n - 1)\ div\ 2$
using *totient-less-not-prime*[*OF* $\langle \neg prime\ n \rangle \langle 1 < n \rangle \langle odd\ n \rangle$]

by (*auto simp add: order-eq elim!: oddE*)

finally have $card\ H < (n - 1)\ div\ 2$.

{

{ fix a
assume $1 \leq a < n$
hence $a \in \{1..<n\}$ **by** *simp*

assume *coprime a n*
then have *coprime a p*
unfolding n **by** *simp*

assume $[a \wedge (2 \wedge (j - 1) * m) \neq 1] \pmod{n}$
hence $[a \wedge m \neq 1] \pmod{n}$
by (*metis dvd-trans dvd-triv-right ord-divides*)

moreover assume *strong-fermat-liar a n*

ultimately obtain i **where** $i \in \{0 ..<k\}$ $[a \wedge (2 \wedge i * m) = n - 1] \pmod{n}$

unfolding *strong-fermat-liar-def* **using** $\langle \text{odd } m \rangle$ *n-less* **by** *blast*

then have $[a^{(2^i * m)} = n - 1] \pmod{p}$
unfolding *n* **using** *cong-modulus-mult-nat* **by** *blast*

moreover have $[n - 1 \neq 1] \pmod{p}$
proof(*subst cong-altdef-nat, goal-cases*)
case *1*
then show *?case* **using** $\langle 1 < n \rangle$ **by** *linarith*
next
case *2*
have $\neg p \text{ dvd } 2$ **using** $\langle 2 < p \rangle$ **by** (*simp add: nat-dvd-not-less*)

moreover have $2 \leq n$ **using** $\langle 1 < n \rangle$ **by** *linarith*

moreover have $p \text{ dvd } n$ **using** *n* **by** *simp*

ultimately have $\neg p \text{ dvd } n - 2$ **using** *dvd-diffD1* **by** *blast*

then show *?case* **by** (*simp add: numeral-2-eq-2*)
qed

ultimately have $[a^{(2^i * m)} \neq \text{Suc } 0] \pmod{p}$ **using** *cong-sym* **by**
(*simp add: cong-def*)

then have $i < j$ **using** *j-least[OF <coprime a p>, of i]* **by** *force*

have $[(a^{(2^{\text{Suc } i} * m)}) = 1] \pmod{n}$
using *square-minus-one-cong-one[OF <1 < n> <a^(2^i * m) = n-1>(mod n)>]*
by (*simp add: power2-eq-square power-mult power-mult-distrib*)

{ assume $i < j - 1$

have $(2 :: \text{nat})^{(j - \text{Suc } 0)} = ((2^i) * 2^{(j - \text{Suc } i)})$
unfolding *power-add[symmetric]* **using** $\langle i < j - 1 \rangle$ **by** *simp*

then have $[a^{(2^{(j-1)} * m)} = (a^{(2^i * m)})^{(2^{(j-1-i)})}] \pmod{n}$
by (*auto intro!: cong-pow-I simp flip: power-mult simp add: algebra-simps power-add*)

also note $\langle [a^{(2^i * m)} = n - 1] \pmod{n} \rangle$

also have $[(n - 1)^{(2^{(j-1-i)})} = 1] \pmod{n}$
using $\langle 1 < n \rangle \langle i < j - 1 \rangle$ **using** *even-pow-cong* **by** *auto*

finally have *False*
using $\langle [a^{(2^{(j-1)} * m)} \neq 1] \pmod{n} \rangle$

} **by** *blast*
 }
hence $i = j - 1$ **using** $\langle i < j \rangle$ **by** *fastforce*
hence $[a \wedge (2 \wedge (j - 1) * m) = n - 1] \pmod n$ **using** $\langle [a \wedge (2 \wedge i * m) = n - 1] \pmod n \rangle$ **by** *simp*
 }
hence $\{a \in \{1..<n\}. \text{strong-fermat-liar } a \ n\} \subseteq H$
using *strong-fermat-liar-imp-fermat-liar*[of - n, OF - $\langle 1 < n \rangle$] *liar-imp-coprime*
by (*auto simp: H-def*)
 }
moreover have *finite H unfolding H-def by auto*
ultimately have *strong-fermat-liar-bounded: card* $\{a \in \{1..<n\}. \text{strong-fermat-liar } a \ n\} < (n - 1) \text{ div } 2$
using *card-mono*[of H] *le-less-trans*[OF - $\langle \text{card } H < (n - 1) \text{ div } 2 \rangle$] **by** *blast*
moreover {
have $\{1..<n\} - \{a \in \{1..<n\}. \text{strong-fermat-liar } a \ n\} = \{a \in \{1..<n\}. \text{strong-fermat-witness } a \ n\}$
strong-fermat-witness $a \ n$
using *strong-fermat-witness-def* **by** *blast*
then have $\text{card } \{a \in \{1..<n\}. \text{strong-fermat-witness } a \ n\} = (n - 1) - \text{card } \{a \in \{1..<n\}. \text{strong-fermat-liar } a \ n\}$
using *card-Diff-subset*[of $\{a \in \{1..<n\}. \text{strong-fermat-liar } a \ n\} \{1..<n\}$]
by *fastforce*
 }
ultimately show $(n - 1) \text{ div } 2 < \text{card } \{a \in \{1..<n\}. \text{strong-fermat-witness } a \ n\}$
by *linarith*
show $\text{real } (\text{card } \{a . 2 \leq a \wedge a < n \wedge \text{strong-fermat-liar } a \ n\}) < \text{real } (n - 2) / 2$
using *strong-fermat-liar-bounded ignore-one one-is-strong-fermat-liar* $\langle 1 < n \rangle$
by *simp*
qed
corollary *strong-fermat-witness-lower-bound:*
assumes *odd n n > 2 ¬prime n*
shows $\text{card } \{a . 2 \leq a \wedge a < n \wedge \text{strong-fermat-liar } a \ n\} < \text{real } (n - 2) / 2$
using *Carmichael-number-imp-lower-bound-on-strong-fermat-witness*(2)[of n]
not-Carmichael-number-imp-card-fermat-witness-bound(2)[of n] *assms*
by (*cases Carmichael-number n*) *auto*
end

9 A Generic View on Probabilistic Prime Tests

theory *Generalized-Primality-Test*

imports

HOL-Probability.Probability

Algebraic-Auxiliaries

begin

definition *primality-test* :: $(\text{nat} \Rightarrow \text{nat} \Rightarrow \text{bool}) \Rightarrow \text{nat} \Rightarrow \text{bool}$ pmf **where**

primality-test P n =

(if n < 3 \vee even n then return-pmf (n = 2) else

do {

a \leftarrow pmf-of-set {2.. n };

return-pmf (P n a)

})

lemma *expectation-of-bool-is-pmf*: *measure-pmf.expectation* M of-bool = pmf M

True

by (*simp add: integral-measure-pmf-real*[**where** A=UNIV] UNIV-bool)

lemma *eq-bernoulli-pmfI*:

assumes pmf p *True* = x

shows p = *bernoulli-pmf* x

proof (*intro pmf-eqI*)

fix b :: bool

from *assms* **have** x \in {0..1} **by** (*auto simp: pmf-le-1*)

thus pmf p b = pmf (*bernoulli-pmf* x) b

using *assms* **by** (*cases b*) (*auto simp: pmf-False-conv-True*)

qed

We require a probabilistic primality test to never classify a prime as composite. It may, however, mistakenly classify composites as primes.

locale *prob-primality-test* =

fixes P :: $\text{nat} \Rightarrow \text{nat} \Rightarrow \text{bool}$ **and** n :: nat

assumes *P-works*: odd n \implies $2 \leq a \implies a < n \implies$ *prime* n \implies P n a

begin

lemma *FalseD*:

assumes *false*: *False* \in *set-pmf* (*primality-test* P n)

shows \neg *prime* n

proof –

from *false* **consider** n \neq 2 n < 3 | n \neq 2 even n |

a **where** \neg P n a odd n $2 \leq a$ a < n

by (*auto simp: primality-test-def not-less split: if-splits*)

then show *?thesis* **proof** *cases*

case 1

then show *?thesis*

```

    by (cases rule: linorder-neqE-nat) (use prime-ge-2-nat[of n] in auto)
  next
    case 2
    then show ?thesis using primes-dvd-imp-eq two-is-prime-nat by blast
  next
    case 3
    then show ?thesis using P-works by blast
qed
qed

```

theorem *prime*:

```

  assumes odd-prime: prime n
  shows primality-test P n = return-pmf True
proof -
  have set-pmf (primality-test P n)  $\subseteq$  {True, False}
    by auto
  moreover from assms have False  $\notin$  set-pmf (primality-test P n)
    using FalseD by auto
  ultimately have set-pmf (primality-test P n)  $\subseteq$  {True}
    by auto
  thus ?thesis
    by (subst (asm) set-pmf-subset-singleton)
qed

```

end

We call a primality test q -good for a fixed positive real number q if the probability that it mistakenly classifies a composite as a prime is less than q .

```

locale good-prob-primality-test = prob-primality-test +
  fixes q :: real
  assumes q-pos: q > 0
  assumes composite-witness-bound:
     $\neg$ prime n  $\implies$  2 < n  $\implies$  odd n  $\implies$ 
      real (card {a . 2  $\leq$  a  $\wedge$  a < n  $\wedge$  P n a}) < q * real (n - 2)
begin

```

lemma *composite-aux*:

```

  assumes  $\neg$ prime n
  shows measure-pmf.expectation (primality-test P n) of-bool < q
  unfolding primality-test-def using assms composite-witness-bound q-pos
  by (clarsimp simp add: pmf-expectation-bind[where A = {2.. $n$ }] sum-of-bool-eq-card
field-simps Int-def
    simp flip: sum-divide-distrib)

```

theorem *composite*:

```

  assumes  $\neg$ prime n
  shows pmf (primality-test P n) True < q
  using composite-aux[OF assms] by (simp add: expectation-of-bool-is-pmf)

```

end

end

10 Fermat's Test

theory *Fermat-Test*

imports

Fermat-Witness

Generalized-Primality-Test

begin

definition *fermat-test* = *primality-test* ($\lambda n a.$ *fermat-liar* *a* *n*)

The Fermat test is a good probabilistic primality test on non-Carmichael numbers.

locale *fermat-test-not-Carmichael-number* =

fixes *n* :: *nat*

assumes *not-Carmichael-number*: \neg *Carmichael-number* *n* \vee *n* < 3

begin

sublocale *fermat-test*: *good-prob-primality-test* $\lambda a n.$ *fermat-liar* *n* *a* *n* 1 / 2

rewrites *primality-test* ($\lambda a n.$ *fermat-liar* *n* *a*) = *fermat-test*

proof –

show *good-prob-primality-test* ($\lambda a n.$ *fermat-liar* *n* *a*) *n* (1 / 2)

using *not-Carmichael-number* *not-Carmichael-number-imp-card-fermat-witness-bound*(3)[*of* *n*]

prime-imp-fermat-liar[*of* *n*]

by *unfold-locales auto*

qed (*auto simp: fermat-test-def*)

end

lemma *not-coprime-imp-fermat-witness*:

fixes *n* :: *nat*

assumes *n* > 1 \neg *coprime* *a* *n*

shows *fermat-witness* *a* *n*

using *assms lucas-coprime-lemma*[*of* *n* – 1 *a* *n*]

by (*auto simp: fermat-witness-def*)

theorem *fermat-test-prime*:

assumes *prime* *n*

shows *fermat-test* *n* = *return-pmf* *True*

proof –

interpret *fermat-test-not-Carmichael-number* *n*

using *assms Carmichael-number-not-prime* **by** *unfold-locales auto*

from *assms* **show** *?thesis* **by** (*rule fermat-test.prime*)

qed

theorem *fermat-test-composite*:

assumes $\neg \text{prime } n \ \neg \text{Carmichael-number } n \ \vee \ n < 3$

shows $\text{pmf } (\text{fermat-test } n) \ \text{True} < 1 / 2$

proof –

interpret *fermat-test-not-Carmichael-number* n **by** *unfold-locales fact+*

from *assms(1)* **show** *?thesis* **by** (*rule fermap-test.composite*)

qed

For a Carmichael number n , Fermat’s test as defined above mistakenly returns ‘True’ with probability $(\varphi(n) - 1)/(n - 2)$. This probability is close to 1 if n has few and big prime factors; it is not quite as bad if it has many and/or small factors, but in that case, simple trial division can also detect compositeness.

Moreover, Fermat’s test only succeeds for a Carmichael number if it happens to guess a number that is not coprime to n . In that case, the fact that we have found a number between 2 and n that is not coprime to n alone is proof that n is composite, and indeed we can even find a non-trivial factor by computing the GCD. This means that for Carmichael numbers, Fermat’s test is essentially no better than the very crude method of attempting to guess numbers coprime to n .

This means that, in general, Fermat’s test is not very helpful for Carmichael numbers.

theorem *fermat-test-Carmichael-number*:

assumes *Carmichael-number* n

shows $\text{fermat-test } n = \text{bernoulli-pmf } (\text{real } (\text{totient } n - 1) / \text{real } (n - 2))$

proof (*rule eq-bernoulli-pmfI*)

from *assms* **have** $n : n > 3 \ \text{odd } n$

using *Carmichael-number-odd Carmichael-number-gt-3* **by** *auto*

from n **have** $\text{fermat-test } n = \text{pmf-of-set } \{2..<n\} \ggg (\lambda a. \text{return-pmf } (\text{fermat-liar } a \ n))$

by (*simp add: fermap-test-def primality-test-def*)

also have $\dots = \text{pmf-of-set } \{2..<n\} \ggg (\lambda a. \text{return-pmf } (\text{coprime } a \ n))$

using n *assms lucas-coprime-lemma[of $n - 1 - n$]*

by (*intro bind-pmf-cong refl*) (*auto simp: Carmichael-number-def fermap-liar-def*)

also have $\text{pmf } \dots \ \text{True} = (\sum_{a=2..<n} \text{indicat-real } \{\text{True}\} (\text{coprime } a \ n)) / \text{real } (n - 2)$

using n **by** (*auto simp: pmf-bind-pmf-of-set*)

also have $(\sum_{a=2..<n} \text{indicat-real } \{\text{True}\} (\text{coprime } a \ n)) = (\sum_{a \mid a \in \{2..<n\} \wedge \text{coprime } a \ n} 1)$

by (*intro sum.mono-neutral-cong-right*) *auto*

also have $\dots = \text{card } \{a \in \{2..<n\}. \text{coprime } a \ n\}$

by *simp*

also have $\{a \in \{2..<n\}. \text{coprime } a \ n\} = \text{totatives } n - \{1\}$

using n **by** (*auto simp: totatives-def order.strict-iff-order[of - n]*)

also have $\text{card } \dots = \text{totient } n - 1$

using n **by** (*subst card-Diff-subset*) (*auto simp: totient-def*)

```

    finally show pmf (fermat-test n) True = real (totient n - 1) / real (n - 2)
      using n by simp
qed

end

```

11 The Miller–Rabin Test

```

theory Miller-Rabin-Test
imports
  Fermat-Witness
  Generalized-Primality-Test
begin

definition miller-rabin = primality-test (λn a. strong-fermat-liar a n)

The test is actually  $\frac{1}{4}$  good, but we only show  $\frac{1}{2}$ , since the former is much
more involved.

interpretation miller-rabin: good-prob-primality-test λn a. strong-fermat-liar a n
n 1 / 2
  rewrites primality-test (λn a. strong-fermat-liar a n) = miller-rabin
proof –
  show good-prob-primality-test (λn a. strong-fermat-liar a n) n (1 / 2)
    by standard (use strong-fermat-witness-lower-bound prime-imp-strong-fermat-witness
in auto)
qed (simp-all add: miller-rabin-def)

end

```

12 The Solovay–Strassen Test

```

theory Solovay-Strassen-Test
imports
  Generalized-Primality-Test
  Euler-Witness
begin

definition solovay-strassen-witness :: nat ⇒ nat ⇒ bool where
  solovay-strassen-witness n a =
    (let x = Jacobi (int a) (int n) in x ≠ 0 ∧ [x = int a ^ ((n - 1) div 2)] (mod
n))

definition solovay-strassen :: nat ⇒ bool pmf where
  solovay-strassen = primality-test solovay-strassen-witness

lemma prime-imp-solovay-strassen-witness:
  assumes prime p odd p a ∈ {2..<p}
  shows solovay-strassen-witness p a

```

```

proof –
  have eq: Jacobi a p = Legendre a p
    using prime-p-Jacobi-eq-Legendre assms by simp
  from ⟨prime p⟩ have coprime p a
    by (rule prime-imp-coprime) (use assms in auto)

  show ?thesis unfolding solovay-strassen-witness-def Let-def eq
  proof
    from ⟨coprime p a⟩ and ⟨prime p⟩ show Legendre (int a) (int p) ≠ 0
      by (auto simp: coprime-commute)
    next
      show [Legendre (int a) (int p) = int a ^ ((p - 1) div 2)] (mod int p)
        using assms by (intro euler-criterion) auto
    qed
  qed

lemma card-solovay-strassen-liars-composite:
  fixes n :: nat
  assumes  $\neg$ prime n n > 2 odd n
  shows card {a ∈ {2..<n}. solovay-strassen-witness n a} < (n - 2) div 2
    (is card ?A < -)
  proof –
    interpret euler-witness-context n
      using assms unfolding euler-witness-context-def by simp
    have card H < (n - 1) div 2
      by (intro card-euler-liars-cosets-limit(2) assms)
    also from assms have H = insert 1 ?A
      by (auto simp: solovay-strassen-witness-def Let-def
        euler-witness-def H-def Jacobi-eq-0-iff-not-coprime)
    also have card ... = card ?A + 1
      by (subst card.insert) auto
    finally show card ?A < (n - 2) div 2
      by linarith
  qed

interpretation solovay-strassen: good-prob-primality-test solovay-strassen-witness
n 1 / 2
  rewrites primality-test solovay-strassen-witness = solovay-strassen
  proof –
    show good-prob-primality-test solovay-strassen-witness n (1 / 2)
  proof
    fix n :: nat assume  $\neg$ prime n n > 2 odd n
    thus real (card {a. 2 ≤ a ∧ a < n ∧ solovay-strassen-witness n a}) < (1 / 2)
  * real (n - 2)
    using card-solovay-strassen-liars-composite[of n] by auto
    qed (use prime-imp-solovay-strassen-witness in auto)
  qed (simp-all add: solovay-strassen-def)

end

```

References

- [1] L. W. after a lecture by Prof. Dr. G. Kemper. Skript: Computer algebra, 2015.
- [2] Brilliant.org. Jacobi symbol, 2018. Last accessed 18 November 2018.
- [3] Brilliant.org. Law of quadratic reciprocity, 2018. Last accessed 18 November 2018.
- [4] Brilliant.org. Legendre symbol, 2018. Last accessed 18 November 2018.
- [5] Brilliant.org. Quadratic residues, 2018. Last accessed 18 November 2018.
- [6] R. D. Carmichael. Note on a new number theory function. *Bulletin of the American Mathematical Society*, 16(5):232–238, 1910.
- [7] K. Conrad. The miller–rabin test, 2018. Last accessed 18 November 2018.
- [8] K. Conrad. The solovay–strassen test, 2018. Last accessed 18 November 2018.
- [9] G. Jameson. Carmichael numbers and pseudoprimes, 2010. Last accessed 18 November 2018.
- [10] B. Kleinberg. The miller-rabin randomized primality test, 2010. Last accessed 18 November 2018.
- [11] mathwizard. Solovay–strassen test, 2013. Last accessed 18 November 2018.
- [12] A. Nicolas. Carmichael number square free, 2016. Last accessed 18 November 2018.
- [13] R. Solovay and V. Strassen. A fast monte-carlo test for primality. *SIAM journal on Computing*, 6(1):84–85, 1977.
- [14] wikipedia. Carmichael number, 2013. Last accessed 18 November 2018.
- [15] wikipedia. Solovay–strassen primality test, 2013. Last accessed 18 November 2018.