

Some results of number theory

Jeremy Avigad
David Gray
Adam Kramer
Thomas M Rasmussen

June 8, 2008

Abstract

This is a collection of formalized proofs of many results of number theory. The proofs of the Chinese Remainder Theorem and Wilson's Theorem are due to Rasmussen. The proof of Gauss's law of quadratic reciprocity is due to Avigad, Gray and Kramer. Proofs can be found in most introductory number theory textbooks; Goldman's *The Queen of Mathematics: a Historically Motivated Guide to Number Theory* provides some historical context.

Avigad, Gray and Kramer have also provided library theories dealing with finite sets and finite sums, divisibility and congruences, parity and residues. The authors are engaged in redesigning and polishing these theories for more serious use. For the latest information in this respect, please see the web page <http://www.andrew.cmu.edu/~avigad/isabelle>. Other theories contain proofs of Euler's criteria, Gauss' lemma, and the law of quadratic reciprocity. The formalization follows Eisenstein's proof, which is the one most commonly found in introductory textbooks; in particular, it follows the presentation in Niven and Zuckerman, *The Theory of Numbers*.

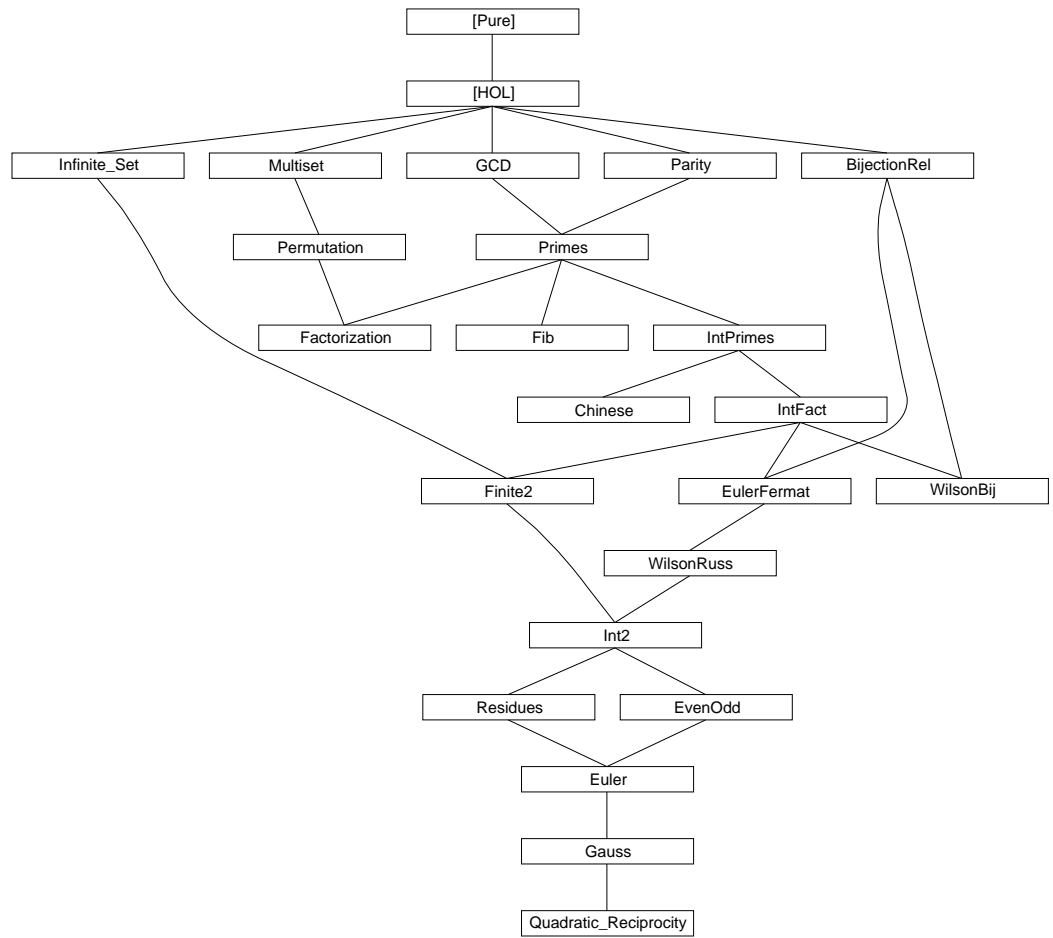
To avoid having to count roots of polynomials, however, we relied on a trick previously used by David Russinoff in formalizing quadratic reciprocity for the Boyer-Moore theorem prover; see Russinoff, David, "A mechanical proof of quadratic reciprocity," *Journal of Automated Reasoning* 8:3-21, 1992. We are grateful to Larry Paulson for calling our attention to this reference.

Contents

1	The Fibonacci function	5
2	Fundamental Theorem of Arithmetic (unique factorization into primes)	6
2.1	Definitions	6
2.2	Arithmetic	7

2.3	Prime list and product	7
2.4	Sorting	8
2.5	Permutation	8
2.6	Existence	9
2.7	Uniqueness	9
3	Divisibility and prime numbers (on integers)	10
3.1	Definitions	11
3.2	Euclid's Algorithm and GCD	11
3.3	Congruences	14
3.4	Modulo	16
3.5	Extended GCD	16
4	The Chinese Remainder Theorem	17
4.1	Definitions	17
4.2	Chinese: uniqueness	19
4.3	Chinese: existence	19
4.4	Chinese	20
5	Bijections between sets	20
6	Factorial on integers	22
7	Fermat's Little Theorem extended to Euler's Totient function	23
7.1	Definitions and lemmas	23
7.2	Fermat	27
8	Wilson's Theorem according to Russinoff	27
8.1	Definitions and lemmas	27
8.2	Wilson	30
9	Wilson's Theorem using a more abstract approach	30
9.1	Definitions and lemmas	30
9.2	Wilson	32
10	Finite Sets and Finite Sums	32
10.1	Useful properties of sums and products	32
10.2	Cardinality of explicit finite sets	33
10.3	Cardinality of finite cartesian products	34
11	Integers: Divisibility and Congruences	34
11.1	Useful lemmas about dvd and powers	34
11.2	Useful properties of congruences	35
11.3	Some properties of MultInv	36

12 Residue Sets	38
12.1 Some useful properties of StandardRes	38
12.2 Relations between StandardRes, SRStar, and SR	39
12.3 Properties relating ResSets with StandardRes	40
12.4 Property for SRStar	40
13 Parity: Even and Odd Integers	41
13.1 Some useful properties about even and odd	41
14 Euler's criterion	43
14.1 Property for MultInvPair	43
14.2 Properties of SetS	44
15 Gauss' Lemma	46
15.1 Basic properties of p	47
15.2 Basic Properties of the Gauss Sets	47
15.3 Relationships Between Gauss Sets	49
15.4 Gauss' Lemma	50
16 The law of Quadratic reciprocity	50
16.1 Stuff about S, S1 and S2	51



1 The Fibonacci function

theory *Fib* **imports** *Primes* **begin**

Fibonacci numbers: proofs of laws taken from: R. L. Graham, D. E. Knuth, O. Patashnik. Concrete Mathematics. (Addison-Wesley, 1989)

```
fun fib :: nat  $\Rightarrow$  nat
where
    fib 0 = 0
  | fib (Suc 0) = 1
  | fib-2: fib (Suc (Suc n)) = fib n + fib (Suc n)
```

The difficulty in these proofs is to ensure that the induction hypotheses are applied before the definition of *fib*. Towards this end, the *fib* equations are not declared to the Simplifier and are applied very selectively at first.

We disable *fib.fib-2fib-2* for simplification ...

```
declare fib-2 [simp del]
```

...then prove a version that has a more restrictive pattern.

```
lemma fib-Suc3: fib (Suc (Suc (Suc n))) = fib (Suc n) + fib (Suc (Suc n))
  <proof>
```

Concrete Mathematics, page 280

```
lemma fib-add: fib (Suc (n + k)) = fib (Suc k) * fib (Suc n) + fib k * fib n
  <proof>
```

```
lemma fib-Suc-neq-0: fib (Suc n)  $\neq$  0
  <proof>
```

```
lemma fib-Suc-gr-0: 0 < fib (Suc n)
  <proof>
```

```
lemma fib-gr-0: 0 < n  $\implies$  0 < fib n
  <proof>
```

Concrete Mathematics, page 278: Cassini's identity. The proof is much easier using integers, not natural numbers!

```
lemma fib-Cassini-int:
  int (fib (Suc (Suc n)) * fib n) =
    (if n mod 2 = 0 then int (fib (Suc n) * fib (Suc n)) - 1
     else int (fib (Suc n) * fib (Suc n)) + 1)
  <proof>
```

We now obtain a version for the natural numbers via the coercion function *int*.

theorem *fib-Cassini*:

$\text{fib } (\text{Suc } (\text{Suc } n)) * \text{fib } n =$
 $(\text{if } n \bmod 2 = 0 \text{ then } \text{fib } (\text{Suc } n) * \text{fib } (\text{Suc } n) - 1$
 $\text{else } \text{fib } (\text{Suc } n) * \text{fib } (\text{Suc } n) + 1)$
 $\langle \text{proof} \rangle$

Toward Law 6.111 of Concrete Mathematics

lemma *gcd-fib-Suc-eq-1*: $\text{gcd } (\text{fib } n, \text{fib } (\text{Suc } n)) = \text{fib } 0$
 $\langle \text{proof} \rangle$

lemma *gcd-fib-add*: $\text{gcd } (\text{fib } m, \text{fib } (n + m)) = \text{gcd } (\text{fib } m, \text{fib } n)$
 $\langle \text{proof} \rangle$

lemma *gcd-fib-diff*: $m \leq n \implies \text{gcd } (\text{fib } m, \text{fib } (n - m)) = \text{gcd } (\text{fib } m, \text{fib } n)$
 $\langle \text{proof} \rangle$

lemma *gcd-fib-mod*: $0 < m \implies \text{gcd } (\text{fib } m, \text{fib } (n \bmod m)) = \text{gcd } (\text{fib } m, \text{fib } n)$
 $\langle \text{proof} \rangle$

lemma *fib-gcd*: $\text{fib } (\text{gcd } (m, n)) = \text{gcd } (\text{fib } m, \text{fib } n)$ — Law 6.111
 $\langle \text{proof} \rangle$

theorem *fib-mult-eq-setsum*:

$\text{fib } (\text{Suc } n) * \text{fib } n = (\sum k \in \{..n\}. \text{fib } k * \text{fib } k)$
 $\langle \text{proof} \rangle$

end

2 Fundamental Theorem of Arithmetic (unique factorization into primes)

theory *Factorization* **imports** *Primes Permutation* **begin**

2.1 Definitions

definition

$\text{primel} :: \text{nat list} \Rightarrow \text{bool}$ **where**
 $\text{primel } xs = (\forall p \in \text{set } xs. \text{prime } p)$

consts

$\text{nondec} :: \text{nat list} \Rightarrow \text{bool}$
 $\text{prod} :: \text{nat list} \Rightarrow \text{nat}$
 $\text{oinsert} :: \text{nat} \Rightarrow \text{nat list} \Rightarrow \text{nat list}$
 $\text{sort} :: \text{nat list} \Rightarrow \text{nat list}$

primrec

$\text{nondec } [] = \text{True}$

$nondec\ (x \# xs) = (case\ xs\ of\ [] \Rightarrow True \mid y \# ys \Rightarrow x \leq y \wedge nondec\ xs)$

primrec

$prod\ [] = Suc\ 0$
 $prod\ (x \# xs) = x * prod\ xs$

primrec

$oinset\ x\ [] = [x]$
 $oinset\ x\ (y \# ys) = (if\ x \leq y\ then\ x \# y \# ys\ else\ y \# oinset\ x\ ys)$

primrec

$sort\ [] = []$
 $sort\ (x \# xs) = oinset\ x\ (sort\ xs)$

2.2 Arithmetic

lemma *one-less-m*: $(m::nat) \neq m * k \Rightarrow m \neq Suc\ 0 \Rightarrow Suc\ 0 < m$
 $\langle proof \rangle$

lemma *one-less-k*: $(m::nat) \neq m * k \Rightarrow Suc\ 0 < m * k \Rightarrow Suc\ 0 < k$
 $\langle proof \rangle$

lemma *mult-left-cancel*: $(0::nat) < k \Rightarrow k * n = k * m \Rightarrow n = m$
 $\langle proof \rangle$

lemma *mn-eq-m-one*: $(0::nat) < m \Rightarrow m * n = m \Rightarrow n = Suc\ 0$
 $\langle proof \rangle$

lemma *prod-mn-less-k*:
 $(0::nat) < n \Rightarrow 0 < k \Rightarrow Suc\ 0 < m \Rightarrow m * n = k \Rightarrow n < k$
 $\langle proof \rangle$

2.3 Prime list and product

lemma *prod-append*: $prod\ (xs @ ys) = prod\ xs * prod\ ys$
 $\langle proof \rangle$

lemma *prod-xy-prod*:

$prod\ (x \# xs) = prod\ (y \# ys) \Rightarrow x * prod\ xs = y * prod\ ys$
 $\langle proof \rangle$

lemma *primel-append*: $primel\ (xs @ ys) = (primel\ xs \wedge primel\ ys)$
 $\langle proof \rangle$

lemma *prime-primel*: $prime\ n \Rightarrow primel\ [n] \wedge prod\ [n] = n$
 $\langle proof \rangle$

lemma *prime-nd-one*: $prime\ p \Rightarrow \neg p\ dvd\ Suc\ 0$
 $\langle proof \rangle$

lemma *hd-dvd-prod*: $\text{prod } (x \# xs) = \text{prod } ys \implies x \text{ dvd } (\text{prod } ys)$
 ⟨proof⟩

lemma *primel-tl*: $\text{primel } (x \# xs) \implies \text{primel } xs$
 ⟨proof⟩

lemma *primel-hd-tl*: $(\text{primel } (x \# xs)) = (\text{prime } x \wedge \text{primel } xs)$
 ⟨proof⟩

lemma *primes-eq*: $\text{prime } p \implies \text{prime } q \implies p \text{ dvd } q \implies p = q$
 ⟨proof⟩

lemma *primel-one-empty*: $\text{primel } xs \implies \text{prod } xs = \text{Suc } 0 \implies xs = []$
 ⟨proof⟩

lemma *prime-g-one*: $\text{prime } p \implies \text{Suc } 0 < p$
 ⟨proof⟩

lemma *prime-g-zero*: $\text{prime } p \implies 0 < p$
 ⟨proof⟩

lemma *primel-nempty-g-one*:
 $\text{primel } xs \implies xs \neq [] \implies \text{Suc } 0 < \text{prod } xs$
 ⟨proof⟩

lemma *primel-prod-gz*: $\text{primel } xs \implies 0 < \text{prod } xs$
 ⟨proof⟩

2.4 Sorting

lemma *nondec-oinsert*: $\text{nondec } xs \implies \text{nondec } (\text{oinsert } x \ xs)$
 ⟨proof⟩

lemma *nondec-sort*: $\text{nondec } (\text{sort } xs)$
 ⟨proof⟩

lemma *x-less-y-oinsert*: $x \leq y \implies l = y \# ys \implies x \# l = \text{oinsert } x \ l$
 ⟨proof⟩

lemma *nondec-sort-eq* [rule-format]: $\text{nondec } xs \longrightarrow xs = \text{sort } xs$
 ⟨proof⟩

lemma *oinsert-x-y*: $\text{oinsert } x \ (\text{oinsert } y \ l) = \text{oinsert } y \ (\text{oinsert } x \ l)$
 ⟨proof⟩

2.5 Permutation

lemma *perm-primel* [rule-format]: $xs <\sim\sim> ys \implies \text{primel } xs \dashrightarrow \text{primel } ys$
 ⟨proof⟩

lemma *perm-prod*: $xs <\sim\sim> ys \implies \text{prod } xs = \text{prod } ys$
 ⟨proof⟩

lemma *perm-subst-oininsert*: $xs <\sim\sim> ys \implies \text{oininsert } a \ xs <\sim\sim> \text{oininsert } a \ ys$
 ⟨proof⟩

lemma *perm-oininsert*: $x \# xs <\sim\sim> \text{oininsert } x \ xs$
 ⟨proof⟩

lemma *perm-sort*: $xs <\sim\sim> \text{sort } xs$
 ⟨proof⟩

lemma *perm-sort-eq*: $xs <\sim\sim> ys \implies \text{sort } xs = \text{sort } ys$
 ⟨proof⟩

2.6 Existence

lemma *ex-nondec-lemma*:
 $\text{primel } xs \implies \exists ys. \text{primel } ys \wedge \text{nondec } ys \wedge \text{prod } ys = \text{prod } xs$
 ⟨proof⟩

lemma *not-prime-ex-mk*:
 $\text{Suc } 0 < n \wedge \neg \text{prime } n \implies$
 $\exists m \ k. \text{Suc } 0 < m \wedge \text{Suc } 0 < k \wedge m < n \wedge k < n \wedge n = m * k$
 ⟨proof⟩

lemma *split-primel*:
 $\text{primel } xs \implies \text{primel } ys \implies \exists l. \text{primel } l \wedge \text{prod } l = \text{prod } xs * \text{prod } ys$
 ⟨proof⟩

lemma *factor-exists* [rule-format]: $\text{Suc } 0 < n \dashrightarrow (\exists l. \text{primel } l \wedge \text{prod } l = n)$
 ⟨proof⟩

lemma *nondec-factor-exists*: $\text{Suc } 0 < n \implies \exists l. \text{primel } l \wedge \text{nondec } l \wedge \text{prod } l = n$
 ⟨proof⟩

2.7 Uniqueness

lemma *prime-dvd-mult-list* [rule-format]:
 $\text{prime } p \implies p \text{ dvd } (\text{prod } xs) \dashrightarrow (\exists m. m : \text{set } xs \wedge p \text{ dvd } m)$
 ⟨proof⟩

lemma *hd-xs-dvd-prod*:
 $\text{primel } (x \# xs) \implies \text{primel } ys \implies \text{prod } (x \# xs) = \text{prod } ys$
 $\implies \exists m. m \in \text{set } ys \wedge x \text{ dvd } m$
 ⟨proof⟩

lemma *prime-dvd-eq*: $\text{primel } (x \# xs) \implies \text{primel } ys \implies m \in \text{set } ys \implies x \text{ dvd } m \implies x = m$

$\langle \text{proof} \rangle$

lemma *hd-xs-eq-prod*:

$\text{primel } (x \# xs) ==>$

$\text{primel } ys ==> \text{prod } (x \# xs) = \text{prod } ys ==> x \in \text{set } ys$

$\langle \text{proof} \rangle$

lemma *perm-primel-ex*:

$\text{primel } (x \# xs) ==>$

$\text{primel } ys ==> \text{prod } (x \# xs) = \text{prod } ys ==> \exists l. ys <\sim\sim> (x \# l)$

$\langle \text{proof} \rangle$

lemma *primel-prod-less*:

$\text{primel } (x \# xs) ==>$

$\text{primel } ys ==> \text{prod } (x \# xs) = \text{prod } ys ==> \text{prod } xs < \text{prod } ys$

$\langle \text{proof} \rangle$

lemma *prod-one-empty*:

$\text{primel } xs ==> p * \text{prod } xs = p ==> \text{prime } p ==> xs = []$

$\langle \text{proof} \rangle$

lemma *uniq-ex-aux*:

$\forall m. m < \text{prod } ys \longrightarrow (\forall xs \ ys. \text{primel } xs \wedge \text{primel } ys \wedge$

$\text{prod } xs = \text{prod } ys \wedge \text{prod } xs = m \longrightarrow xs <\sim\sim> ys) ==>$

$\text{primel } list ==> \text{primel } x ==> \text{prod } list = \text{prod } x ==> \text{prod } x < \text{prod } ys$

$==> x <\sim\sim> list$

$\langle \text{proof} \rangle$

lemma *factor-unique* [rule-format]:

$\forall xs \ ys. \text{primel } xs \wedge \text{primel } ys \wedge \text{prod } xs = \text{prod } ys \wedge \text{prod } xs = n$

$\longrightarrow xs <\sim\sim> ys$

$\langle \text{proof} \rangle$

lemma *perm-nondec-unique*:

$xs <\sim\sim> ys ==> \text{nondec } xs ==> \text{nondec } ys ==> xs = ys$

$\langle \text{proof} \rangle$

theorem *unique-prime-factorization* [rule-format]:

$\forall n. \text{Suc } 0 < n \longrightarrow (\exists ! l. \text{primel } l \wedge \text{nondec } l \wedge \text{prod } l = n)$

$\langle \text{proof} \rangle$

end

3 Divisibility and prime numbers (on integers)

theory *IntPrimes*

imports *Primes*

begin

The *dvd* relation, GCD, Euclid's extended algorithm, primes, congruences (all on the Integers). Comparable to theory *Primes*, but *dvd* is included here as it is not present in main HOL. Also includes extended GCD and congruences not present in *Primes*.

3.1 Definitions

consts

xzgcd :: *int* * *int* * *int* * *int* * *int* * *int* * *int* * *int* ==> *int* * *int* * *int*

recdef *xzgcd*

measure (($\lambda(m, n, r', r, s', s, t', t). \text{nat } r$)
 :: *int* * *int* * *int* * *int* * *int* * *int* * *int* * *int* ==> *nat*)
xzgcd (*m*, *n*, *r'*, *r*, *s'*, *s*, *t'*, *t*) =
 (if *r* ≤ 0 then (*r'*, *s'*, *t'*)
 else *xzgcd* (*m*, *n*, *r*, *r' mod r*,
s, *s' - (r' div r) * s*,
t, *t' - (r' div r) * t*))

definition

zgcd :: *int* * *int* ==> *int* **where**
zgcd = ($\lambda(x, y). \text{int } (\text{gcd } (\text{nat } (\text{abs } x), \text{nat } (\text{abs } y)))$)

definition

zprime :: *int* ==> *bool* **where**
zprime *p* = (*1* < *p* ∧ (∀ *m*. 0 <= *m* & *m dvd p* ==> *m* = 1 ∨ *m* = *p*))

definition

xzgcd :: *int* ==> *int* ==> *int* * *int* * *int* **where**
xzgcd *m* *n* = *xzgcd* (*m*, *n*, *m*, *n*, 1, 0, 0, 1)

definition

zcong :: *int* ==> *int* ==> *int* ==> *bool* ((1[- = -] '(mod -))) **where**
 [*a* = *b*] (mod *m*) = (*m dvd* (*a* - *b*))

gcd lemmas

lemma *gcd-add1-eq*: *gcd* (*m* + *k*, *k*) = *gcd* (*m* + *k*, *m*)
 <proof>

lemma *gcd-diff2*: *m* ≤ *n* ==> *gcd* (*n*, *n* - *m*) = *gcd* (*n*, *m*)
 <proof>

3.2 Euclid's Algorithm and GCD

lemma *zgcd-0* [*simp*]: *zgcd* (*m*, 0) = *abs m*
 <proof>

lemma *zgcd-0-left* [*simp*]: *zgcd* (0, *m*) = *abs m*

$\langle proof \rangle$

lemma *zgcd-zminus* [simp]: $zgcd (-m, n) = zgcd (m, n)$
 $\langle proof \rangle$

lemma *zgcd-zminus2* [simp]: $zgcd (m, -n) = zgcd (m, n)$
 $\langle proof \rangle$

lemma *zgcd-non-0*: $0 < n \implies zgcd (m, n) = zgcd (n, m \bmod n)$
 $\langle proof \rangle$

lemma *zgcd-eq*: $zgcd (m, n) = zgcd (n, m \bmod n)$
 $\langle proof \rangle$

lemma *zgcd-1* [simp]: $zgcd (m, 1) = 1$
 $\langle proof \rangle$

lemma *zgcd-0-1-iff* [simp]: $(zgcd (0, m) = 1) = (abs\ m = 1)$
 $\langle proof \rangle$

lemma *zgcd-zdvd1* [iff]: $zgcd (m, n) \text{ dvd } m$
 $\langle proof \rangle$

lemma *zgcd-zdvd2* [iff]: $zgcd (m, n) \text{ dvd } n$
 $\langle proof \rangle$

lemma *zgcd-greatest-iff*: $k \text{ dvd } zgcd (m, n) = (k \text{ dvd } m \wedge k \text{ dvd } n)$
 $\langle proof \rangle$

lemma *zgcd-commute*: $zgcd (m, n) = zgcd (n, m)$
 $\langle proof \rangle$

lemma *zgcd-1-left* [simp]: $zgcd (1, m) = 1$
 $\langle proof \rangle$

lemma *zgcd-assoc*: $zgcd (zgcd (k, m), n) = zgcd (k, zgcd (m, n))$
 $\langle proof \rangle$

lemma *zgcd-left-commute*: $zgcd (k, zgcd (m, n)) = zgcd (m, zgcd (k, n))$
 $\langle proof \rangle$

lemmas *zgcd-ac* = *zgcd-assoc* *zgcd-commute* *zgcd-left-commute*
— addition is an AC-operator

lemma *zgcd-zmult-distrib2*: $0 \leq k \implies k * zgcd (m, n) = zgcd (k * m, k * n)$
 $\langle proof \rangle$

lemma *zgcd-zmult-distrib2-abs*: $zgcd (k * m, k * n) = abs\ k * zgcd (m, n)$
 $\langle proof \rangle$

lemma *zgcd-self* [simp]: $0 \leq m \implies \text{zgcd } (m, m) = m$
 ⟨proof⟩

lemma *zgcd-zmult-eq-self* [simp]: $0 \leq k \implies \text{zgcd } (k, k * n) = k$
 ⟨proof⟩

lemma *zgcd-zmult-eq-self2* [simp]: $0 \leq k \implies \text{zgcd } (k * n, k) = k$
 ⟨proof⟩

lemma *zrelprime-zdvd-zmult-aux*:
 $\text{zgcd } (n, k) = 1 \implies k \text{ dvd } m * n \implies 0 \leq m \implies k \text{ dvd } m$
 ⟨proof⟩

lemma *zrelprime-zdvd-zmult*: $\text{zgcd } (n, k) = 1 \implies k \text{ dvd } m * n \implies k \text{ dvd } m$
 ⟨proof⟩

lemma *zgcd-geq-zero*: $0 \leq \text{zgcd}(x, y)$
 ⟨proof⟩

This is merely a sanity check on *zprime*, since the previous version denoted the empty set.

lemma *zprime 2*
 ⟨proof⟩

lemma *zprime-imp-zrelprime*:
 $\text{zprime } p \implies \neg p \text{ dvd } n \implies \text{zgcd } (n, p) = 1$
 ⟨proof⟩

lemma *zless-zprime-imp-zrelprime*:
 $\text{zprime } p \implies 0 < n \implies n < p \implies \text{zgcd } (n, p) = 1$
 ⟨proof⟩

lemma *zprime-zdvd-zmult*:
 $0 \leq (m::\text{int}) \implies \text{zprime } p \implies p \text{ dvd } m * n \implies p \text{ dvd } m \vee p \text{ dvd } n$
 ⟨proof⟩

lemma *zgcd-zadd-zmult* [simp]: $\text{zgcd } (m + n * k, n) = \text{zgcd } (m, n)$
 ⟨proof⟩

lemma *zgcd-zdvd-zgcd-zmult*: $\text{zgcd } (m, n) \text{ dvd } \text{zgcd } (k * m, n)$
 ⟨proof⟩

lemma *zgcd-zmult-zdvd-zgcd*:
 $\text{zgcd } (k, n) = 1 \implies \text{zgcd } (k * m, n) \text{ dvd } \text{zgcd } (m, n)$
 ⟨proof⟩

lemma *zgcd-zmult-cancel*: $\text{zgcd } (k, n) = 1 \implies \text{zgcd } (k * m, n) = \text{zgcd } (m, n)$
 ⟨proof⟩

lemma *zgcd-zgcd-zmult*:

$$\text{zgcd } (k, m) = 1 \implies \text{zgcd } (n, m) = 1 \implies \text{zgcd } (k * n, m) = 1$$

<proof>

lemma *zdvd-iff-zgcd*: $0 < m \implies (m \text{ dvd } n) = (\text{zgcd } (n, m) = m)$

<proof>

3.3 Congruences

lemma *zcong-1* [*simp*]: $[a = b] \text{ (mod } 1)$

<proof>

lemma *zcong-refl* [*simp*]: $[k = k] \text{ (mod } m)$

<proof>

lemma *zcong-sym*: $[a = b] \text{ (mod } m) = [b = a] \text{ (mod } m)$

<proof>

lemma *zcong-zadd*:

$$[a = b] \text{ (mod } m) \implies [c = d] \text{ (mod } m) \implies [a + c = b + d] \text{ (mod } m)$$

<proof>

lemma *zcong-zdiff*:

$$[a = b] \text{ (mod } m) \implies [c = d] \text{ (mod } m) \implies [a - c = b - d] \text{ (mod } m)$$

<proof>

lemma *zcong-trans*:

$$[a = b] \text{ (mod } m) \implies [b = c] \text{ (mod } m) \implies [a = c] \text{ (mod } m)$$

<proof>

lemma *zcong-zmult*:

$$[a = b] \text{ (mod } m) \implies [c = d] \text{ (mod } m) \implies [a * c = b * d] \text{ (mod } m)$$

<proof>

lemma *zcong-scalar*: $[a = b] \text{ (mod } m) \implies [a * k = b * k] \text{ (mod } m)$

<proof>

lemma *zcong-scalar2*: $[a = b] \text{ (mod } m) \implies [k * a = k * b] \text{ (mod } m)$

<proof>

lemma *zcong-zmult-self*: $[a * m = b * m] \text{ (mod } m)$

<proof>

lemma *zcong-square*:

$$[| \text{zprime } p; \ 0 < a; \ [a * a = 1] \text{ (mod } p) |] \implies [a = 1] \text{ (mod } p) \vee [a = p - 1] \text{ (mod } p)$$

<proof>

lemma *zcong-cancel*:

$0 \leq m \implies$
 $zgcd(k, m) = 1 \implies [a * k = b * k] \text{ (mod } m) = [a = b] \text{ (mod } m)$
 $\langle \text{proof} \rangle$

lemma *zcong-cancel2*:

$0 \leq m \implies$
 $zgcd(k, m) = 1 \implies [k * a = k * b] \text{ (mod } m) = [a = b] \text{ (mod } m)$
 $\langle \text{proof} \rangle$

lemma *zcong-zgcd-zmult-zmod*:

$[a = b] \text{ (mod } m) \implies [a = b] \text{ (mod } n) \implies zgcd(m, n) = 1$
 $\implies [a = b] \text{ (mod } m * n)$
 $\langle \text{proof} \rangle$

lemma *zcong-zless-imp-eq*:

$0 \leq a \implies$
 $a < m \implies 0 \leq b \implies b < m \implies [a = b] \text{ (mod } m) \implies a = b$
 $\langle \text{proof} \rangle$

lemma *zcong-square-zless*:

$zprime\ p \implies 0 < a \implies a < p \implies$
 $[a * a = 1] \text{ (mod } p) \implies a = 1 \vee a = p - 1$
 $\langle \text{proof} \rangle$

lemma *zcong-not*:

$0 < a \implies a < m \implies 0 < b \implies b < a \implies \neg [a = b] \text{ (mod } m)$
 $\langle \text{proof} \rangle$

lemma *zcong-zless-0*:

$0 \leq a \implies a < m \implies [a = 0] \text{ (mod } m) \implies a = 0$
 $\langle \text{proof} \rangle$

lemma *zcong-zless-unique*:

$0 < m \implies (\exists! b. 0 \leq b \wedge b < m \wedge [a = b] \text{ (mod } m))$
 $\langle \text{proof} \rangle$

lemma *zcong-iff-lin*: $([a = b] \text{ (mod } m)) = (\exists k. b = a + m * k)$

$\langle \text{proof} \rangle$

lemma *zgcd-zcong-zgcd*:

$0 < m \implies$
 $zgcd(a, m) = 1 \implies [a = b] \text{ (mod } m) \implies zgcd(b, m) = 1$
 $\langle \text{proof} \rangle$

lemma *zcong-zmod-aux*:

$a - b = (m :: int) * (a \text{ div } m - b \text{ div } m) + (a \text{ mod } m - b \text{ mod } m)$
 $\langle \text{proof} \rangle$

lemma *zcong-zmod*: $[a = b] \text{ (mod } m) = [a \text{ mod } m = b \text{ mod } m] \text{ (mod } m)$
 $\langle \text{proof} \rangle$

lemma *zcong-zmod-eq*: $0 < m \implies [a = b] \text{ (mod } m) = (a \text{ mod } m = b \text{ mod } m)$
 $\langle \text{proof} \rangle$

lemma *zcong-zminus* [iff]: $[a = b] \text{ (mod } -m) = [a = b] \text{ (mod } m)$
 $\langle \text{proof} \rangle$

lemma *zcong-zero* [iff]: $[a = b] \text{ (mod } 0) = (a = b)$
 $\langle \text{proof} \rangle$

lemma $[a = b] \text{ (mod } m) = (a \text{ mod } m = b \text{ mod } m)$
 $\langle \text{proof} \rangle$

3.4 Modulo

lemma *zmod-zdvd-zmod*:
 $0 < (m::int) \implies m \text{ dvd } b \implies (a \text{ mod } b \text{ mod } m) = (a \text{ mod } m)$
 $\langle \text{proof} \rangle$

3.5 Extended GCD

declare *xzgcd.simps* [simp del]

lemma *xzgcd-correct-aux1*:
 $zgcd \text{ (} r', r) = k \dashrightarrow 0 < r \dashrightarrow$
 $(\exists sn \text{ } tn. \text{ } xzgcd \text{ (} m, n, r', r, s', s, t', t) = (k, sn, tn))$
 $\langle \text{proof} \rangle$

lemma *xzgcd-correct-aux2*:
 $(\exists sn \text{ } tn. \text{ } xzgcd \text{ (} m, n, r', r, s', s, t', t) = (k, sn, tn)) \dashrightarrow 0 < r \dashrightarrow$
 $zgcd \text{ (} r', r) = k$
 $\langle \text{proof} \rangle$

lemma *xzgcd-correct*:
 $0 < n \implies (zgcd \text{ (} m, n) = k) = (\exists s \text{ } t. \text{ } xzgcd \text{ } m \text{ } n = (k, s, t))$
 $\langle \text{proof} \rangle$

xzgcd linear

lemma *xzgcd-linear-aux1*:
 $(a - r * b) * m + (c - r * d) * (n::int) =$
 $(a * m + c * n) - r * (b * m + d * n)$
 $\langle \text{proof} \rangle$

lemma *xzgcd-linear-aux2*:
 $r' = s' * m + t' * n \implies r = s * m + t * n$
 $\implies (r' \text{ mod } r) = (s' - (r' \text{ div } r) * s) * m + (t' - (r' \text{ div } r) * t) * (n::int)$
 $\langle \text{proof} \rangle$

lemma *order-le-neq-implies-less*: $(x::'a::\text{order}) \leq y \implies x \neq y \implies x < y$
 <proof>

lemma *xzgca-linear* [rule-format]:
 $0 < r \dashv\vdash \text{xzgca } (m, n, r', r, s', s, t', t) = (rn, sn, tn) \dashv\vdash$
 $r' = s' * m + t' * n \dashv\vdash r = s * m + t * n \dashv\vdash rn = sn * m + tn * n$
 <proof>

lemma *xzgcd-linear*:
 $0 < n \implies \text{xzgcd } m \ n = (r, s, t) \implies r = s * m + t * n$
 <proof>

lemma *zgcd-ex-linear*:
 $0 < n \implies \text{zgcd } (m, n) = k \implies (\exists s \ t. k = s * m + t * n)$
 <proof>

lemma *zcong-lineq-ex*:
 $0 < n \implies \text{zgcd } (a, n) = 1 \implies \exists x. [a * x = 1] \ (\text{mod } n)$
 <proof>

lemma *zcong-lineq-unique*:
 $0 < n \implies$
 $\text{zgcd } (a, n) = 1 \implies \exists! x. 0 \leq x \wedge x < n \wedge [a * x = b] \ (\text{mod } n)$
 <proof>

end

4 The Chinese Remainder Theorem

theory *Chinese* imports *IntPrimes* begin

The Chinese Remainder Theorem for an arbitrary finite number of equations. (The one-equation case is included in theory *IntPrimes*. Uses functions for indexing.¹

4.1 Definitions

consts
 $\text{funprod} :: (\text{nat} \Rightarrow \text{int}) \Rightarrow \text{nat} \Rightarrow \text{nat} \Rightarrow \text{int}$
 $\text{funsum} :: (\text{nat} \Rightarrow \text{int}) \Rightarrow \text{nat} \Rightarrow \text{nat} \Rightarrow \text{int}$

primrec
 $\text{funprod } f \ i \ 0 = f \ i$
 $\text{funprod } f \ i \ (\text{Suc } n) = f \ (\text{Suc } (i + n)) * \text{funprod } f \ i \ n$

¹Maybe *funprod* and *funsum* should be based on general *fold* on indices?

primrec

$\text{funsum } f \ i \ 0 = f \ i$
 $\text{funsum } f \ i \ (\text{Suc } n) = f \ (\text{Suc } (i + n)) + \text{funsum } f \ i \ n$

definition

$m\text{-cond} :: \text{nat} \Rightarrow (\text{nat} \Rightarrow \text{int}) \Rightarrow \text{bool}$ **where**
 $m\text{-cond } n \ mf =$
 $((\forall i. i \leq n \longrightarrow 0 < mf \ i) \wedge$
 $(\forall i \ j. i \leq n \wedge j \leq n \wedge i \neq j \longrightarrow \text{zgcd } (mf \ i, mf \ j) = 1))$

definition

$km\text{-cond} :: \text{nat} \Rightarrow (\text{nat} \Rightarrow \text{int}) \Rightarrow (\text{nat} \Rightarrow \text{int}) \Rightarrow \text{bool}$ **where**
 $km\text{-cond } n \ kf \ mf = (\forall i. i \leq n \longrightarrow \text{zgcd } (kf \ i, mf \ i) = 1)$

definition

$lincong\text{-sol} ::$
 $\text{nat} \Rightarrow (\text{nat} \Rightarrow \text{int}) \Rightarrow (\text{nat} \Rightarrow \text{int}) \Rightarrow (\text{nat} \Rightarrow \text{int}) \Rightarrow \text{int} \Rightarrow \text{bool}$
where
 $lincong\text{-sol } n \ kf \ bf \ mf \ x = (\forall i. i \leq n \longrightarrow \text{zcong } (kf \ i * x) (bf \ i) (mf \ i))$

definition

$mhf :: (\text{nat} \Rightarrow \text{int}) \Rightarrow \text{nat} \Rightarrow \text{nat} \Rightarrow \text{int}$ **where**
 $mhf \ mf \ n \ i =$
 $(\text{if } i = 0 \text{ then funprod } mf \ (\text{Suc } 0) \ (n - \text{Suc } 0)$
 $\text{else if } i = n \text{ then funprod } mf \ 0 \ (n - \text{Suc } 0)$
 $\text{else funprod } mf \ 0 \ (i - \text{Suc } 0) * \text{funprod } mf \ (\text{Suc } i) \ (n - \text{Suc } 0 - i))$

definition

$xilin\text{-sol} ::$
 $\text{nat} \Rightarrow \text{nat} \Rightarrow (\text{nat} \Rightarrow \text{int}) \Rightarrow (\text{nat} \Rightarrow \text{int}) \Rightarrow (\text{nat} \Rightarrow \text{int}) \Rightarrow \text{int}$
where
 $xilin\text{-sol } i \ n \ kf \ bf \ mf =$
 $(\text{if } 0 < n \wedge i \leq n \wedge m\text{-cond } n \ mf \wedge km\text{-cond } n \ kf \ mf \text{ then}$
 $(\text{SOME } x. 0 \leq x \wedge x < mf \ i \wedge \text{zcong } (kf \ i * mhf \ mf \ n \ i * x) (bf \ i) (mf \ i))$
 $\text{else } 0)$

definition

$x\text{-sol} :: \text{nat} \Rightarrow (\text{nat} \Rightarrow \text{int}) \Rightarrow (\text{nat} \Rightarrow \text{int}) \Rightarrow (\text{nat} \Rightarrow \text{int}) \Rightarrow \text{int}$ **where**
 $x\text{-sol } n \ kf \ bf \ mf = \text{funsum } (\lambda i. xilin\text{-sol } i \ n \ kf \ bf \ mf * mhf \ mf \ n \ i) \ 0 \ n$

funprod and funsum

lemma *funprod-pos*: $(\forall i. i \leq n \longrightarrow 0 < mf \ i) \implies 0 < \text{funprod } mf \ 0 \ n$
 $\langle \text{proof} \rangle$

lemma *funprod-zgcd* [rule-format (no-asm)]:

$(\forall i. k \leq i \wedge i \leq k + l \longrightarrow \text{zgcd } (mf \ i, mf \ m) = 1) \longrightarrow$
 $\text{zgcd } (\text{funprod } mf \ k \ l, mf \ m) = 1$
 $\langle \text{proof} \rangle$

lemma *funprod-zdvd* [rule-format]:

$$k \leq i \longrightarrow i \leq k + l \longrightarrow mf\ i\ dvd\ funprod\ mf\ k\ l$$

<proof>

lemma *funsum-mod*:

$$funsum\ f\ k\ l\ mod\ m = funsum\ (\lambda i. (f\ i)\ mod\ m)\ k\ l\ mod\ m$$

<proof>

lemma *funsum-zero* [rule-format (no-asm)]:

$$(\forall i. k \leq i \wedge i \leq k + l \longrightarrow f\ i = 0) \longrightarrow (funsum\ f\ k\ l) = 0$$

<proof>

lemma *funsum-oneelem* [rule-format (no-asm)]:

$$k \leq j \longrightarrow j \leq k + l \longrightarrow$$

$$(\forall i. k \leq i \wedge i \leq k + l \wedge i \neq j \longrightarrow f\ i = 0) \longrightarrow$$

$$funsum\ f\ k\ l = f\ j$$

<proof>

4.2 Chinese: uniqueness

lemma *zcong-funprod-aux*:

$$m\text{-}cond\ n\ mf \implies km\text{-}cond\ n\ kf\ mf$$

$$\implies lincong\text{-}sol\ n\ kf\ bf\ mf\ x \implies lincong\text{-}sol\ n\ kf\ bf\ mf\ y$$

$$\implies [x = y]\ (mod\ mf\ n)$$

<proof>

lemma *zcong-funprod* [rule-format]:

$$m\text{-}cond\ n\ mf \longrightarrow km\text{-}cond\ n\ kf\ mf \longrightarrow$$

$$lincong\text{-}sol\ n\ kf\ bf\ mf\ x \longrightarrow lincong\text{-}sol\ n\ kf\ bf\ mf\ y \longrightarrow$$

$$[x = y]\ (mod\ funprod\ mf\ 0\ n)$$

<proof>

4.3 Chinese: existence

lemma *unique-xi-sol*:

$$0 < n \implies i \leq n \implies m\text{-}cond\ n\ mf \implies km\text{-}cond\ n\ kf\ mf$$

$$\implies \exists! x. 0 \leq x \wedge x < mf\ i \wedge [kf\ i * mh\ mf\ n\ i * x = bf\ i]\ (mod\ mf\ i)$$

<proof>

lemma *x-sol-lin-aux*:

$$0 < n \implies i \leq n \implies j \leq n \implies j \neq i \implies mf\ j\ dvd\ mh\ mf\ n\ i$$

<proof>

lemma *x-sol-lin*:

$$0 < n \implies i \leq n$$

$$\implies x\text{-}sol\ n\ kf\ bf\ mf\ mod\ mf\ i =$$

$$xilin\text{-}sol\ i\ n\ kf\ bf\ mf * mh\ mf\ n\ i\ mod\ mf\ i$$

<proof>

4.4 Chinese

lemma *chinese-remainder*:

$0 < n \implies m\text{-cond } n \text{ } mf \implies km\text{-cond } n \text{ } kf \text{ } mf$
 $\implies \exists !x. 0 \leq x \wedge x < \text{funprod } mf \text{ } 0 \text{ } n \wedge \text{lincong-sol } n \text{ } kf \text{ } bf \text{ } mf \text{ } x$
 $\langle \text{proof} \rangle$

end

5 Bijections between sets

theory *BijectionRel* **imports** *Main* **begin**

Inductive definitions of bijections between two different sets and between the same set. Theorem for relating the two definitions.

inductive-set

$\text{bijR} :: ('a \Rightarrow 'b \Rightarrow \text{bool}) \Rightarrow ('a \text{ set} * 'b \text{ set}) \text{ set}$
for $P :: 'a \Rightarrow 'b \Rightarrow \text{bool}$

where

$\text{empty } [\text{simp}]: (\{\}, \{\}) \in \text{bijR } P$
 $|\text{insert}: P \text{ } a \text{ } b \implies a \notin A \implies b \notin B \implies (A, B) \in \text{bijR } P$
 $\implies (\text{insert } a \text{ } A, \text{insert } b \text{ } B) \in \text{bijR } P$

Add extra condition to *insert*: $\forall b \in B. \neg P \text{ } a \text{ } b$ (and similar for *A*).

definition

$\text{bijP} :: ('a \Rightarrow 'a \Rightarrow \text{bool}) \Rightarrow 'a \text{ set} \Rightarrow \text{bool}$ **where**
 $\text{bijP } P \text{ } F = (\forall a \text{ } b. a \in F \wedge P \text{ } a \text{ } b \longrightarrow b \in F)$

definition

$\text{uniqP} :: ('a \Rightarrow 'a \Rightarrow \text{bool}) \Rightarrow \text{bool}$ **where**
 $\text{uniqP } P = (\forall a \text{ } b \text{ } c \text{ } d. P \text{ } a \text{ } b \wedge P \text{ } c \text{ } d \longrightarrow (a = c) = (b = d))$

definition

$\text{symP} :: ('a \Rightarrow 'a \Rightarrow \text{bool}) \Rightarrow \text{bool}$ **where**
 $\text{symP } P = (\forall a \text{ } b. P \text{ } a \text{ } b = P \text{ } b \text{ } a)$

inductive-set

$\text{bijER} :: ('a \Rightarrow 'a \Rightarrow \text{bool}) \Rightarrow 'a \text{ set set}$
for $P :: 'a \Rightarrow 'a \Rightarrow \text{bool}$

where

$\text{empty } [\text{simp}]: \{\} \in \text{bijER } P$
 $|\text{insert1}: P \text{ } a \text{ } a \implies a \notin A \implies A \in \text{bijER } P \implies \text{insert } a \text{ } A \in \text{bijER } P$
 $|\text{insert2}: P \text{ } a \text{ } b \implies a \neq b \implies a \notin A \implies b \notin A \implies A \in \text{bijER } P$
 $\implies \text{insert } a \text{ } (\text{insert } b \text{ } A) \in \text{bijER } P$

bijR

lemma *fin-bijRl*: $(A, B) \in \text{bijR } P \implies \text{finite } A$
 ⟨proof⟩

lemma *fin-bijRr*: $(A, B) \in \text{bijR } P \implies \text{finite } B$
 ⟨proof⟩

lemma *aux-induct*:
 assumes *major*: $\text{finite } F$
 and *subs*: $F \subseteq A$
 and *cases*: $P \{\}$
 !! $F a. F \subseteq A \implies a \in A \implies a \notin F \implies P F \implies P (\text{insert } a F)$
 shows $P F$
 ⟨proof⟩

lemma *inj-func-bijR-aux1*:
 $A \subseteq B \implies a \notin A \implies a \in B \implies \text{inj-on } f B \implies f a \notin f ' A$
 ⟨proof⟩

lemma *inj-func-bijR-aux2*:
 $\forall a. a \in A \dashrightarrow P a (f a) \implies \text{inj-on } f A \implies \text{finite } A \implies F \leq A$
 $\implies (F, f ' F) \in \text{bijR } P$
 ⟨proof⟩

lemma *inj-func-bijR*:
 $\forall a. a \in A \dashrightarrow P a (f a) \implies \text{inj-on } f A \implies \text{finite } A$
 $\implies (A, f ' A) \in \text{bijR } P$
 ⟨proof⟩

bijER

lemma *fin-bijER*: $A \in \text{bijER } P \implies \text{finite } A$
 ⟨proof⟩

lemma *aux1*:
 $a \notin A \implies a \notin B \implies F \subseteq \text{insert } a A \implies F \subseteq \text{insert } a B \implies a \in F$
 $\implies \exists C. F = \text{insert } a C \wedge a \notin C \wedge C \leq A \wedge C \leq B$
 ⟨proof⟩

lemma *aux2*: $a \neq b \implies a \notin A \implies b \notin B \implies a \in F \implies b \in F$
 $\implies F \subseteq \text{insert } a A \implies F \subseteq \text{insert } b B$
 $\implies \exists C. F = \text{insert } a (\text{insert } b C) \wedge a \notin C \wedge b \notin C \wedge C \subseteq A \wedge C \subseteq B$
 ⟨proof⟩

lemma *aux-uniq*: $\text{uniqP } P \implies P a b \implies P c d \implies (a = c) = (b = d)$
 ⟨proof⟩

lemma *aux-sym*: $\text{symP } P \implies P a b = P b a$
 ⟨proof⟩

lemma *aux-in1*:

$uniqP\ P ==> b \notin C ==> P\ b\ b ==> bijP\ P\ (insert\ b\ C) ==> bijP\ P\ C$
 $\langle proof \rangle$

lemma *aux-in2*:

$symP\ P ==> uniqP\ P ==> a \notin C ==> b \notin C ==> a \neq b ==> P\ a\ b$
 $==> bijP\ P\ (insert\ a\ (insert\ b\ C)) ==> bijP\ P\ C$
 $\langle proof \rangle$

lemma *aux-foo*: $\forall a\ b. Q\ a \wedge P\ a\ b \dashrightarrow R\ b ==> P\ a\ b ==> Q\ a ==> R\ b$

$\langle proof \rangle$

lemma *aux-bij*: $bijP\ P\ F ==> symP\ P ==> P\ a\ b ==> (a \in F) = (b \in F)$

$\langle proof \rangle$

lemma *aux-bijRER*:

$(A, B) \in bijR\ P ==> uniqP\ P ==> symP\ P$
 $==> \forall F. bijP\ P\ F \wedge F \subseteq A \wedge F \subseteq B \dashrightarrow F \in bijER\ P$
 $\langle proof \rangle$

lemma *bijR-bijER*:

$(A, A) \in bijR\ P ==>$
 $bijP\ P\ A ==> uniqP\ P ==> symP\ P ==> A \in bijER\ P$
 $\langle proof \rangle$

end

6 Factorial on integers

theory *IntFact* **imports** *IntPrimes* **begin**

Factorial on integers and recursively defined set including all Integers from 2 up to a . Plus definition of product of finite set.

consts

$zfact :: int \Rightarrow int$
 $d22set :: int \Rightarrow int\ set$

recdef *zfact* *measure* $((\lambda n. nat\ n) :: int \Rightarrow nat)$

$zfact\ n = (if\ n \leq 0\ then\ 1\ else\ n * zfact\ (n - 1))$

recdef *d22set* *measure* $((\lambda a. nat\ a) :: int \Rightarrow nat)$

$d22set\ a = (if\ 1 < a\ then\ insert\ a\ (d22set\ (a - 1))\ else\ \{\})$

d22set — recursively defined set including all integers from 2 up to a

declare *d22set.simps* [*simp del*]

```

lemma d22set-induct:
  assumes !!a.  $P \ \{\}$  a
    and !!a.  $1 < (a::int) \implies P \ (d22set \ (a - 1)) \ (a - 1) \implies P \ (d22set \ a) \ a$ 
  shows  $P \ (d22set \ u) \ u$ 
  <proof>

lemma d22set-g-1 [rule-format]:  $b \in d22set \ a \dashrightarrow 1 < b$ 
  <proof>

lemma d22set-le [rule-format]:  $b \in d22set \ a \dashrightarrow b \leq a$ 
  <proof>

lemma d22set-le-swap:  $a < b \implies b \notin d22set \ a$ 
  <proof>

lemma d22set-mem:  $1 < b \implies b \leq a \implies b \in d22set \ a$ 
  <proof>

lemma d22set-fin: finite (d22set a)
  <proof>

declare zfact.simps [simp del]

lemma d22set-prod-zfact:  $\prod (d22set \ a) = zfact \ a$ 
  <proof>

end

```

7 Fermat's Little Theorem extended to Euler's Totient function

theory *EulerFermat* **imports** *BijectionRel IntFact* **begin**

Fermat's Little Theorem extended to Euler's Totient function. More abstract approach than Boyer-Moore (which seems necessary to achieve the extended version).

7.1 Definitions and lemmas

```

inductive-set
  RsetR :: int => int set set
  for m :: int
  where
    empty [simp]:  $\{\} \in RsetR \ m$ 

```

| *insert*: $A \in RsetR\ m \implies zgcd\ (a, m) = 1 \implies$
 $\forall a'. a' \in A \dashv\dashv \neg\ zcong\ a\ a'\ m \implies insert\ a\ A \in RsetR\ m$

consts

BnorRset :: $int * int \Rightarrow int\ set$

recdef *BnorRset*

measure $((\lambda(a, m). nat\ a) :: int * int \Rightarrow nat)$
BnorRset $(a, m) =$
(if $0 < a$ *then*
let $na = BnorRset\ (a - 1, m)$
in *(if* $zgcd\ (a, m) = 1$ *then* *insert* $a\ na$ *else* na
else $\{\}$)

definition

norRRset :: $int \Rightarrow int\ set$ **where**
norRRset $m = BnorRset\ (m - 1, m)$

definition

noXRRset :: $int \Rightarrow int \Rightarrow int\ set$ **where**
noXRRset $m\ x = (\lambda a. a * x)\ 'norRRset\ m$

definition

phi :: $int \Rightarrow nat$ **where**
phi $m = card\ (norRRset\ m)$

definition

is-RRset :: $int\ set \Rightarrow int \Rightarrow bool$ **where**
is-RRset $A\ m = (A \in RsetR\ m \wedge card\ A = phi\ m)$

definition

RRset2norRR :: $int\ set \Rightarrow int \Rightarrow int \Rightarrow int$ **where**
RRset2norRR $A\ m\ a =$
(if $1 < m \wedge is-RRset\ A\ m \wedge a \in A$ *then*
SOME $b. zcong\ a\ b\ m \wedge b \in norRRset\ m$
else 0)

definition

zcongm :: $int \Rightarrow int \Rightarrow int \Rightarrow bool$ **where**
zcongm $m = (\lambda a\ b. zcong\ a\ b\ m)$

lemma *abs-eq-1-iff* [*iff*]: $(abs\ z = (1::int)) = (z = 1 \vee z = -1)$

— LCP: not sure why this lemma is needed now

<proof>

norRRset

declare *BnorRset.simps* [*simp del*]

lemma *BnorRset-induct*:

assumes $!!a\ m.\ P\ \{\}$ $a\ m$
and $!!a\ m.\ 0 < (a::int) \implies P\ (BnorRset\ (a - 1,\ m::int))\ (a - 1)\ m$
 $\implies P\ (BnorRset(a,m))\ a\ m$
shows $P\ (BnorRset(u,v))\ u\ v$
 $\langle proof \rangle$

lemma *Bnor-mem-zle* [rule-format]: $b \in BnorRset\ (a,\ m) \longrightarrow b \leq a$
 $\langle proof \rangle$

lemma *Bnor-mem-zle-swap*: $a < b \implies b \notin BnorRset\ (a,\ m)$
 $\langle proof \rangle$

lemma *Bnor-mem-zg* [rule-format]: $b \in BnorRset\ (a,\ m) \dashrightarrow 0 < b$
 $\langle proof \rangle$

lemma *Bnor-mem-if* [rule-format]:
 $zgcd\ (b,\ m) = 1 \dashrightarrow 0 < b \dashrightarrow b \leq a \dashrightarrow b \in BnorRset\ (a,\ m)$
 $\langle proof \rangle$

lemma *Bnor-in-RsetR* [rule-format]: $a < m \dashrightarrow BnorRset\ (a,\ m) \in RsetR\ m$
 $\langle proof \rangle$

lemma *Bnor-fin*: $finite\ (BnorRset\ (a,\ m))$
 $\langle proof \rangle$

lemma *norR-mem-unique-aux*: $a \leq b - 1 \implies a < (b::int)$
 $\langle proof \rangle$

lemma *norR-mem-unique*:
 $1 < m \implies$
 $zgcd\ (a,\ m) = 1 \implies \exists!b.\ [a = b]\ (mod\ m) \wedge b \in norRRset\ m$
 $\langle proof \rangle$

noXRRset

lemma *RRset-gcd* [rule-format]:
 $is-RRset\ A\ m \implies a \in A \dashrightarrow zgcd\ (a,\ m) = 1$
 $\langle proof \rangle$

lemma *RsetR-zmult-mono*:
 $A \in RsetR\ m \implies$
 $0 < m \implies zgcd\ (x,\ m) = 1 \implies (\lambda a.\ a * x) ' A \in RsetR\ m$
 $\langle proof \rangle$

lemma *card-nor-eq-noX*:
 $0 < m \implies$
 $zgcd\ (x,\ m) = 1 \implies card\ (noXRRset\ m\ x) = card\ (norRRset\ m)$
 $\langle proof \rangle$

lemma *noX-is-RRset*:

$0 < m \implies \text{zgcd } (x, m) = 1 \implies \text{is-RRset } (\text{noXRRset } m \ x) \ m$
 $\langle \text{proof} \rangle$

lemma *aux-some*:

$1 < m \implies \text{is-RRset } A \ m \implies a \in A$
 $\implies \text{zcong } a \ (\text{SOME } b. [a = b] \ (\text{mod } m) \wedge b \in \text{norRRset } m) \ m \wedge$
 $(\text{SOME } b. [a = b] \ (\text{mod } m) \wedge b \in \text{norRRset } m) \in \text{norRRset } m$
 $\langle \text{proof} \rangle$

lemma *RRset2norRR-correct*:

$1 < m \implies \text{is-RRset } A \ m \implies a \in A \implies$
 $[a = \text{RRset2norRR } A \ m \ a] \ (\text{mod } m) \wedge \text{RRset2norRR } A \ m \ a \in \text{norRRset } m$
 $\langle \text{proof} \rangle$

lemmas *RRset2norRR-correct1* =

RRset2norRR-correct [THEN conjunct1, standard]

lemmas *RRset2norRR-correct2* =

RRset2norRR-correct [THEN conjunct2, standard]

lemma *RsetR-fin*: $A \in \text{RsetR } m \implies \text{finite } A$

$\langle \text{proof} \rangle$

lemma *RRset-zcong-eq* [rule-format]:

$1 < m \implies$
 $\text{is-RRset } A \ m \implies [a = b] \ (\text{mod } m) \implies a \in A \dashrightarrow b \in A \dashrightarrow a = b$
 $\langle \text{proof} \rangle$

lemma *aux*:

$P \ (\text{SOME } a. P \ a) \implies Q \ (\text{SOME } a. Q \ a) \implies$
 $(\text{SOME } a. P \ a) = (\text{SOME } a. Q \ a) \implies \exists a. P \ a \wedge Q \ a$
 $\langle \text{proof} \rangle$

lemma *RRset2norRR-inj*:

$1 < m \implies \text{is-RRset } A \ m \implies \text{inj-on } (\text{RRset2norRR } A \ m) \ A$
 $\langle \text{proof} \rangle$

lemma *RRset2norRR-eq-norR*:

$1 < m \implies \text{is-RRset } A \ m \implies \text{RRset2norRR } A \ m \ ' A = \text{norRRset } m$
 $\langle \text{proof} \rangle$

lemma *Bnor-prod-power-aux*: $a \notin A \implies \text{inj } f \implies f \ a \notin f \ ' A$

$\langle \text{proof} \rangle$

lemma *Bnor-prod-power* [rule-format]:

$x \neq 0 \implies a < m \dashrightarrow \prod ((\lambda a. a * x) \ ' BnorRset \ (a, m)) =$
 $\prod (BnorRset(a, m)) * x^{\text{card } (BnorRset \ (a, m))}$
 $\langle \text{proof} \rangle$

7.2 Fermat

lemma *bijzcong-zcong-prod*:

$(A, B) \in \text{bijR } (\text{zcong } m) \implies [\prod A = \prod B] \pmod{m}$
 $\langle \text{proof} \rangle$

lemma *Bnor-prod-zgcd* [rule-format]:

$a < m \dashrightarrow \text{zgcd } (\prod (\text{BnorRset}(a, m)), m) = 1$
 $\langle \text{proof} \rangle$

theorem *Euler-Fermat*:

$0 < m \implies \text{zgcd } (x, m) = 1 \implies [x^{\text{phi } m} = 1] \pmod{m}$
 $\langle \text{proof} \rangle$

lemma *Bnor-prime*:

$[\text{zprime } p; a < p] \implies \text{card } (\text{BnorRset } (a, p)) = \text{nat } a$
 $\langle \text{proof} \rangle$

lemma *phi-prime*: $\text{zprime } p \implies \text{phi } p = \text{nat } (p - 1)$

$\langle \text{proof} \rangle$

theorem *Little-Fermat*:

$\text{zprime } p \implies \neg p \text{ dvd } x \implies [x^{\text{nat } (p - 1)} = 1] \pmod{p}$
 $\langle \text{proof} \rangle$

end

8 Wilson's Theorem according to Russinoff

theory *WilsonRuss* **imports** *EulerFermat* **begin**

Wilson's Theorem following quite closely Russinoff's approach using Boyer-Moore (using finite sets instead of lists, though).

8.1 Definitions and lemmas

definition

$\text{inv} :: \text{int} \Rightarrow \text{int} \Rightarrow \text{int}$ **where**
 $\text{inv } p \ a = (a^{\text{nat } (p - 2)}) \pmod{p}$

consts

$\text{wset} :: \text{int} * \text{int} \Rightarrow \text{int set}$

recdef *wset*

$\text{measure } ((\lambda(a, p). \text{nat } a) :: \text{int} * \text{int} \Rightarrow \text{nat})$
 $\text{wset } (a, p) =$
 $(\text{if } 1 < a \text{ then}$
 $\text{let } ws = \text{wset } (a - 1, p)$

in (if a ∈ ws then ws else insert a (insert (inv p a) ws)) else {})

inv

lemma *inv-is-inv-aux*: $1 < m \implies \text{Suc } (\text{nat } (m - 2)) = \text{nat } (m - 1)$
 $\langle \text{proof} \rangle$

lemma *inv-is-inv*:

$\text{zprime } p \implies 0 < a \implies a < p \implies [a * \text{inv } p \ a = 1] \ (\text{mod } p)$
 $\langle \text{proof} \rangle$

lemma *inv-distinct*:

$\text{zprime } p \implies 1 < a \implies a < p - 1 \implies a \neq \text{inv } p \ a$
 $\langle \text{proof} \rangle$

lemma *inv-not-0*:

$\text{zprime } p \implies 1 < a \implies a < p - 1 \implies \text{inv } p \ a \neq 0$
 $\langle \text{proof} \rangle$

lemma *inv-not-1*:

$\text{zprime } p \implies 1 < a \implies a < p - 1 \implies \text{inv } p \ a \neq 1$
 $\langle \text{proof} \rangle$

lemma *inv-not-p-minus-1-aux*:

$[a * (p - 1) = 1] \ (\text{mod } p) = [a = p - 1] \ (\text{mod } p)$
 $\langle \text{proof} \rangle$

lemma *inv-not-p-minus-1*:

$\text{zprime } p \implies 1 < a \implies a < p - 1 \implies \text{inv } p \ a \neq p - 1$
 $\langle \text{proof} \rangle$

lemma *inv-g-1*:

$\text{zprime } p \implies 1 < a \implies a < p - 1 \implies 1 < \text{inv } p \ a$
 $\langle \text{proof} \rangle$

lemma *inv-less-p-minus-1*:

$\text{zprime } p \implies 1 < a \implies a < p - 1 \implies \text{inv } p \ a < p - 1$
 $\langle \text{proof} \rangle$

lemma *inv-inv-aux*: $5 \leq p \implies$

$\text{nat } (p - 2) * \text{nat } (p - 2) = \text{Suc } (\text{nat } (p - 1) * \text{nat } (p - 3))$
 $\langle \text{proof} \rangle$

lemma *zcong-zpower-zmult*:

$[x^y = 1] \ (\text{mod } p) \implies [x^{(y * z)} = 1] \ (\text{mod } p)$
 $\langle \text{proof} \rangle$

lemma *inv-inv*: $\text{zprime } p \implies$

$5 \leq p \implies 0 < a \implies a < p \implies \text{inv } p \ (\text{inv } p \ a) = a$
 $\langle \text{proof} \rangle$

wset

declare *wset.simps* [*simp del*]

lemma *wset-induct*:

assumes $!!a\ p.\ P\ \{\}\ a\ p$
and $!!a\ p.\ 1 < (a::int) \implies$
 $P\ (wset\ (a - 1, p))\ (a - 1)\ p \implies P\ (wset\ (a, p))\ a\ p$
shows $P\ (wset\ (u, v))\ u\ v$
<proof>

lemma *wset-mem-imp-or* [*rule-format*]:

$1 < a \implies b \notin wset\ (a - 1, p)$
 $\implies b \in wset\ (a, p) \dashrightarrow b = a \vee b = inv\ p\ a$
<proof>

lemma *wset-mem-mem* [*simp*]: $1 < a \implies a \in wset\ (a, p)$

<proof>

lemma *wset-subset*: $1 < a \implies b \in wset\ (a - 1, p) \implies b \in wset\ (a, p)$

<proof>

lemma *wset-g-1* [*rule-format*]:

$zprime\ p \dashrightarrow a < p - 1 \dashrightarrow b \in wset\ (a, p) \dashrightarrow 1 < b$
<proof>

lemma *wset-less* [*rule-format*]:

$zprime\ p \dashrightarrow a < p - 1 \dashrightarrow b \in wset\ (a, p) \dashrightarrow b < p - 1$
<proof>

lemma *wset-mem* [*rule-format*]:

$zprime\ p \dashrightarrow$
 $a < p - 1 \dashrightarrow 1 < b \dashrightarrow b \leq a \dashrightarrow b \in wset\ (a, p)$
<proof>

lemma *wset-mem-inv-mem* [*rule-format*]:

$zprime\ p \dashrightarrow 5 \leq p \dashrightarrow a < p - 1 \dashrightarrow b \in wset\ (a, p)$
 $\dashrightarrow inv\ p\ b \in wset\ (a, p)$
<proof>

lemma *wset-inv-mem-mem*:

$zprime\ p \implies 5 \leq p \implies a < p - 1 \implies 1 < b \implies b < p - 1$
 $\implies inv\ p\ b \in wset\ (a, p) \implies b \in wset\ (a, p)$
<proof>

lemma *wset-fin*: *finite* (*wset* (*a*, *p*))

<proof>

lemma *wset-zcong-prod-1* [*rule-format*]:

$zprime\ p \dashrightarrow$

$5 \leq p \dashv\dashv a < p - 1 \dashv\dashv [(\prod x \in wset(a, p). x) = 1] \pmod{p}$
 $\langle proof \rangle$

lemma *d22set-eq-wset*: $zprime\ p \implies d22set\ (p - 2) = wset\ (p - 2, p)$
 $\langle proof \rangle$

8.2 Wilson

lemma *prime-g-5*: $zprime\ p \implies p \neq 2 \implies p \neq 3 \implies 5 \leq p$
 $\langle proof \rangle$

theorem *Wilson-Russ*:
 $zprime\ p \implies [zfact\ (p - 1) = -1] \pmod{p}$
 $\langle proof \rangle$

end

9 Wilson's Theorem using a more abstract approach

theory *WilsonBij* **imports** *BijectionRel IntFact* **begin**

Wilson's Theorem using a more "abstract" approach based on bijections between sets. Does not use Fermat's Little Theorem (unlike Russinoff).

9.1 Definitions and lemmas

definition
 $reciR :: int \Rightarrow int \Rightarrow int \Rightarrow bool$ **where**
 $reciR\ p = (\lambda a\ b. zcong\ (a * b)\ 1\ p \wedge 1 < a \wedge a < p - 1 \wedge 1 < b \wedge b < p - 1)$

definition
 $inv :: int \Rightarrow int \Rightarrow int$ **where**
 $inv\ p\ a =$
 $(if\ zprime\ p \wedge 0 < a \wedge a < p\ then$
 $(SOME\ x. 0 \leq x \wedge x < p \wedge zcong\ (a * x)\ 1\ p)$
 $else\ 0)$

Inverse

lemma *inv-correct*:
 $zprime\ p \implies 0 < a \implies a < p$
 $\implies 0 \leq inv\ p\ a \wedge inv\ p\ a < p \wedge [a * inv\ p\ a = 1] \pmod{p}$
 $\langle proof \rangle$

lemmas *inv-ge* = *inv-correct* [*THEN* *conjunct1*, *standard*]

lemmas *inv-less* = *inv-correct* [*THEN* *conjunct2*, *THEN* *conjunct1*, *standard*]

lemmas *inv-is-inv* = *inv-correct* [*THEN conjunct2*, *THEN conjunct2*, *standard*]

lemma *inv-not-0*:

zprime p ==> 1 < a ==> a < p - 1 ==> inv p a ≠ 0
 — same as *WilsonRuss*
 ⟨*proof*⟩

lemma *inv-not-1*:

zprime p ==> 1 < a ==> a < p - 1 ==> inv p a ≠ 1
 — same as *WilsonRuss*
 ⟨*proof*⟩

lemma *aux*: $[a * (p - 1) = 1] \text{ (mod } p) = [a = p - 1] \text{ (mod } p)$

— same as *WilsonRuss*
 ⟨*proof*⟩

lemma *inv-not-p-minus-1*:

zprime p ==> 1 < a ==> a < p - 1 ==> inv p a ≠ p - 1
 — same as *WilsonRuss*
 ⟨*proof*⟩

Below is slightly different as we don't expand *inv* but use “*correct*” theorems.

lemma *inv-g-1*: *zprime p ==> 1 < a ==> a < p - 1 ==> 1 < inv p a*

⟨*proof*⟩

lemma *inv-less-p-minus-1*:

zprime p ==> 1 < a ==> a < p - 1 ==> inv p a < p - 1
 — ditto
 ⟨*proof*⟩

Bijection

lemma *aux1*: $1 < x ==> 0 \leq (x::int)$

⟨*proof*⟩

lemma *aux2*: $1 < x ==> 0 < (x::int)$

⟨*proof*⟩

lemma *aux3*: $x \leq p - 2 ==> x < (p::int)$

⟨*proof*⟩

lemma *aux4*: $x \leq p - 2 ==> x < (p::int) - 1$

⟨*proof*⟩

lemma *inv-inj*: *zprime p ==> inj-on (inv p) (d22set (p - 2))*

⟨*proof*⟩

lemma *inv-d22set-d22set*:

zprime p ==> inv p ‘ d22set (p - 2) = d22set (p - 2)
 ⟨*proof*⟩

lemma *d22set-d22set-bij*:
 $zprime\ p ==> (d22set\ (p - 2), d22set\ (p - 2)) \in bijR\ (reciR\ p)$
 $\langle proof \rangle$

lemma *reciP-bijP*: $zprime\ p ==> bijP\ (reciR\ p)\ (d22set\ (p - 2))$
 $\langle proof \rangle$

lemma *reciP-uniq*: $zprime\ p ==> uniqP\ (reciR\ p)$
 $\langle proof \rangle$

lemma *reciP-sym*: $zprime\ p ==> symP\ (reciR\ p)$
 $\langle proof \rangle$

lemma *bijER-d22set*: $zprime\ p ==> d22set\ (p - 2) \in bijER\ (reciR\ p)$
 $\langle proof \rangle$

9.2 Wilson

lemma *bijER-zcong-prod-1*:
 $zprime\ p ==> A \in bijER\ (reciR\ p) ==> [\prod A = 1] \ (mod\ p)$
 $\langle proof \rangle$

theorem *Wilson-Bij*: $zprime\ p ==> [zfact\ (p - 1) = -1] \ (mod\ p)$
 $\langle proof \rangle$

end

10 Finite Sets and Finite Sums

theory *Finite2*
imports *Main IntFact Infinite-Set*
begin

These are useful for combinatorial and number-theoretic counting arguments.

10.1 Useful properties of sums and products

lemma *setsum-same-function-zcong*:
assumes $a: \forall x \in S. [f\ x = g\ x] \ (mod\ m)$
shows $[setsum\ f\ S = setsum\ g\ S] \ (mod\ m)$
 $\langle proof \rangle$

lemma *setprod-same-function-zcong*:
assumes $a: \forall x \in S. [f\ x = g\ x] \ (mod\ m)$
shows $[setprod\ f\ S = setprod\ g\ S] \ (mod\ m)$
 $\langle proof \rangle$

lemma *setsum-const*: $\text{finite } X \implies \text{setsum } (\%x. (c :: \text{int})) X = c * \text{int}(\text{card } X)$
 $\langle \text{proof} \rangle$

lemma *setsum-const2*: $\text{finite } X \implies \text{int}(\text{setsum } (\%x. (c :: \text{nat})) X) =$
 $\text{int}(c) * \text{int}(\text{card } X)$
 $\langle \text{proof} \rangle$

lemma *setsum-const-mult*: $\text{finite } A \implies \text{setsum } (\%x. c * ((f x) :: \text{int})) A =$
 $c * \text{setsum } f A$
 $\langle \text{proof} \rangle$

10.2 Cardinality of explicit finite sets

lemma *finite-surjI*: $[\mid B \subseteq f ` A; \text{finite } A \mid] \implies \text{finite } B$
 $\langle \text{proof} \rangle$

lemma *bdd-nat-set-l-finite*: $\text{finite } \{y :: \text{nat} . y < x\}$
 $\langle \text{proof} \rangle$

lemma *bdd-nat-set-le-finite*: $\text{finite } \{y :: \text{nat} . y \leq x\}$
 $\langle \text{proof} \rangle$

lemma *bdd-int-set-l-finite*: $\text{finite } \{x :: \text{int} . 0 \leq x \ \& \ x < n\}$
 $\langle \text{proof} \rangle$

lemma *bdd-int-set-le-finite*: $\text{finite } \{x :: \text{int} . 0 \leq x \ \& \ x \leq n\}$
 $\langle \text{proof} \rangle$

lemma *bdd-int-set-l-l-finite*: $\text{finite } \{x :: \text{int} . 0 < x \ \& \ x < n\}$
 $\langle \text{proof} \rangle$

lemma *bdd-int-set-l-le-finite*: $\text{finite } \{x :: \text{int} . 0 < x \ \& \ x \leq n\}$
 $\langle \text{proof} \rangle$

lemma *card-bdd-nat-set-l*: $\text{card } \{y :: \text{nat} . y < x\} = x$
 $\langle \text{proof} \rangle$

lemma *card-bdd-nat-set-le*: $\text{card } \{y :: \text{nat} . y \leq x\} = \text{Suc } x$
 $\langle \text{proof} \rangle$

lemma *card-bdd-int-set-l*: $0 \leq (n :: \text{int}) \implies \text{card } \{y. 0 \leq y \ \& \ y < n\} = \text{nat } n$
 $\langle \text{proof} \rangle$

lemma *card-bdd-int-set-le*: $0 \leq (n :: \text{int}) \implies \text{card } \{y. 0 \leq y \ \& \ y \leq n\} =$
 $\text{nat } n + 1$
 $\langle \text{proof} \rangle$

lemma *card-bdd-int-set-l-le*: $0 \leq (n :: \text{int}) \implies$

$\text{card } \{x. 0 < x \ \& \ x \leq n\} = \text{nat } n$
 $\langle \text{proof} \rangle$

lemma *card-bdd-int-set-l-l*: $0 < (n::\text{int}) \implies$
 $\text{card } \{x. 0 < x \ \& \ x < n\} = \text{nat } n - 1$
 $\langle \text{proof} \rangle$

lemma *int-card-bdd-int-set-l-l*: $0 < n \implies$
 $\text{int}(\text{card } \{x. 0 < x \ \& \ x < n\}) = n - 1$
 $\langle \text{proof} \rangle$

lemma *int-card-bdd-int-set-l-le*: $0 \leq n \implies$
 $\text{int}(\text{card } \{x. 0 < x \ \& \ x \leq n\}) = n$
 $\langle \text{proof} \rangle$

10.3 Cardinality of finite cartesian products

Lemmas for counting arguments.

lemma *setsum-bij-eq*: $[| \text{finite } A; \text{finite } B; f ' A \subseteq B; \text{inj-on } f \ A;$
 $g ' B \subseteq A; \text{inj-on } g \ B |] \implies \text{setsum } g \ B = \text{setsum } (g \circ f) \ A$
 $\langle \text{proof} \rangle$

lemma *setprod-bij-eq*: $[| \text{finite } A; \text{finite } B; f ' A \subseteq B; \text{inj-on } f \ A;$
 $g ' B \subseteq A; \text{inj-on } g \ B |] \implies \text{setprod } g \ B = \text{setprod } (g \circ f) \ A$
 $\langle \text{proof} \rangle$

end

11 Integers: Divisibility and Congruences

theory *Int2* **imports** *Finite2 WilsonRuss* **begin**

definition

$\text{MultInv} :: \text{int} \implies \text{int} \implies \text{int}$ **where**
 $\text{MultInv } p \ x = x \wedge \text{nat } (p - 2)$

11.1 Useful lemmas about dvd and powers

lemma *zpower-zdvd-prop1*:
 $0 < n \implies p \ \text{dvd } y \implies p \ \text{dvd } ((y::\text{int}) \wedge^n)$
 $\langle \text{proof} \rangle$

lemma *zdvd-bounds*: $n \ \text{dvd } m \implies m \leq (0::\text{int}) \mid n \leq m$
 $\langle \text{proof} \rangle$

lemma *zprime-zdvd-zmult-better*: $[| \text{zprime } p; \ p \ \text{dvd } (m * n) |] \implies$
 $(p \ \text{dvd } m) \mid (p \ \text{dvd } n)$

$\langle \text{proof} \rangle$

lemma *zpower-zdvd-prop2*:

$\text{zprime } p \implies p \text{ dvd } ((y::\text{int}) \wedge n) \implies 0 < n \implies p \text{ dvd } y$

$\langle \text{proof} \rangle$

lemma *div-prop1*: $[0 < z; (x::\text{int}) < y * z] \implies x \text{ div } z < y$

$\langle \text{proof} \rangle$

lemma *div-prop2*: $[0 < z; (x::\text{int}) < (y * z) + z] \implies x \text{ div } z \leq y$

$\langle \text{proof} \rangle$

lemma *zdiv-leq-prop*: $[0 < y] \implies y * (x \text{ div } y) \leq (x::\text{int})$

$\langle \text{proof} \rangle$

11.2 Useful properties of congruences

lemma *zcong-eq-zdvd-prop*: $[x = 0](\text{mod } p) = (p \text{ dvd } x)$

$\langle \text{proof} \rangle$

lemma *zcong-id*: $[m = 0] (\text{mod } m)$

$\langle \text{proof} \rangle$

lemma *zcong-shift*: $[a = b] (\text{mod } m) \implies [a + c = b + c] (\text{mod } m)$

$\langle \text{proof} \rangle$

lemma *zcong-zpower*: $[x = y](\text{mod } m) \implies [x^z = y^z](\text{mod } m)$

$\langle \text{proof} \rangle$

lemma *zcong-eq-trans*: $[[a = b](\text{mod } m); b = c; [c = d](\text{mod } m)] \implies [a = d](\text{mod } m)$

$\langle \text{proof} \rangle$

lemma *aux1*: $a - b = (c::\text{int}) \implies a = c + b$

$\langle \text{proof} \rangle$

lemma *zcong-zmult-prop1*: $[a = b](\text{mod } m) \implies ([c = a * d](\text{mod } m) = [c = b * d](\text{mod } m))$

$\langle \text{proof} \rangle$

lemma *zcong-zmult-prop2*: $[a = b](\text{mod } m) \implies$

$([c = d * a](\text{mod } m) = [c = d * b](\text{mod } m))$

$\langle \text{proof} \rangle$

lemma *zcong-zmult-prop3*: $[\text{zprime } p; \sim [x = 0] (\text{mod } p);$

$\sim [y = 0] (\text{mod } p)] \implies \sim [x * y = 0] (\text{mod } p)$

$\langle \text{proof} \rangle$

lemma *zcong-less-eq*: $[0 < x; 0 < y; 0 < m; [x = y] (\text{mod } m);$

$x < m; y < m \implies x = y$
 $\langle \text{proof} \rangle$

lemma *zcong-neg-1-impl-ne-1*: $\llbracket 2 < p; [x = -1] \text{ (mod } p) \rrbracket \implies$
 $\sim([x = 1] \text{ (mod } p))$
 $\langle \text{proof} \rangle$

lemma *zcong-zero-equiv-div*: $[a = 0] \text{ (mod } m) = (m \text{ dvd } a)$
 $\langle \text{proof} \rangle$

lemma *zcong-zprime-prod-zero*: $\llbracket \text{zprime } p; 0 < a \rrbracket \implies$
 $[a * b = 0] \text{ (mod } p) \implies [a = 0] \text{ (mod } p) \mid [b = 0] \text{ (mod } p)$
 $\langle \text{proof} \rangle$

lemma *zcong-zprime-prod-zero-contr*: $\llbracket \text{zprime } p; 0 < a \rrbracket \implies$
 $\sim[a = 0] \text{ (mod } p) \ \& \ \sim[b = 0] \text{ (mod } p) \implies \sim[a * b = 0] \text{ (mod } p)$
 $\langle \text{proof} \rangle$

lemma *zcong-not-zero*: $\llbracket 0 < x; x < m \rrbracket \implies \sim[x = 0] \text{ (mod } m)$
 $\langle \text{proof} \rangle$

lemma *zcong-zero*: $\llbracket 0 \leq x; x < m; [x = 0] \text{ (mod } m) \rrbracket \implies x = 0$
 $\langle \text{proof} \rangle$

lemma *all-relprime-prod-relprime*: $\llbracket \text{finite } A; \forall x \in A. (\text{zgcd}(x, y) = 1) \rrbracket$
 $\implies \text{zgcd}(\text{setprod id } A, y) = 1$
 $\langle \text{proof} \rangle$

11.3 Some properties of MultInv

lemma *MultInv-prop1*: $\llbracket 2 < p; [x = y] \text{ (mod } p) \rrbracket \implies$
 $[(\text{MultInv } p \ x) = (\text{MultInv } p \ y)] \text{ (mod } p)$
 $\langle \text{proof} \rangle$

lemma *MultInv-prop2*: $\llbracket 2 < p; \text{zprime } p; \sim([x = 0] \text{ (mod } p)) \rrbracket \implies$
 $[(x * (\text{MultInv } p \ x)) = 1] \text{ (mod } p)$
 $\langle \text{proof} \rangle$

lemma *MultInv-prop2a*: $\llbracket 2 < p; \text{zprime } p; \sim([x = 0] \text{ (mod } p)) \rrbracket \implies$
 $[(\text{MultInv } p \ x) * x = 1] \text{ (mod } p)$
 $\langle \text{proof} \rangle$

lemma *aux-1*: $2 < p \implies ((\text{nat } p) - 2) = (\text{nat } (p - 2))$
 $\langle \text{proof} \rangle$

lemma *aux-2*: $2 < p \implies 0 < \text{nat } (p - 2)$
 $\langle \text{proof} \rangle$

lemma *MultInv-prop3*: $\llbracket 2 < p; \text{zprime } p; \sim([x = 0] \text{ (mod } p)) \rrbracket \implies$

$\sim([MultInv\ p\ x = 0](mod\ p))$
 $\langle proof \rangle$

lemma *aux--1*: $[| \ 2 < p; \ zprime\ p; \ \sim([x = 0](mod\ p)) |] ==>$
 $[(MultInv\ p\ (MultInv\ p\ x)) = (x * (MultInv\ p\ x) * (MultInv\ p\ (MultInv\ p\ x))) (mod\ p)$
 $\langle proof \rangle$

lemma *aux--2*: $[| \ 2 < p; \ zprime\ p; \ \sim([x = 0](mod\ p)) |] ==>$
 $[(x * (MultInv\ p\ x) * (MultInv\ p\ (MultInv\ p\ x))) = x] (mod\ p)$
 $\langle proof \rangle$

lemma *MultInv-prop4*: $[| \ 2 < p; \ zprime\ p; \ \sim([x = 0](mod\ p)) |] ==>$
 $[(MultInv\ p\ (MultInv\ p\ x)) = x] (mod\ p)$
 $\langle proof \rangle$

lemma *MultInv-prop5*: $[| \ 2 < p; \ zprime\ p; \ \sim([x = 0](mod\ p));$
 $\sim([y = 0](mod\ p)); \ [(MultInv\ p\ x) = (MultInv\ p\ y)] (mod\ p) |] ==>$
 $[x = y] (mod\ p)$
 $\langle proof \rangle$

lemma *MultInv-zcong-prop1*: $[| \ 2 < p; \ [j = k] (mod\ p) |] ==>$
 $[a * MultInv\ p\ j = a * MultInv\ p\ k] (mod\ p)$
 $\langle proof \rangle$

lemma *aux---1*: $[j = a * MultInv\ p\ k] (mod\ p) ==>$
 $[j * k = a * MultInv\ p\ k * k] (mod\ p)$
 $\langle proof \rangle$

lemma *aux---2*: $[| 2 < p; \ zprime\ p; \ \sim([k = 0](mod\ p));$
 $[j * k = a * MultInv\ p\ k * k] (mod\ p) |] ==> [j * k = a] (mod\ p)$
 $\langle proof \rangle$

lemma *aux---3*: $[j * k = a] (mod\ p) ==> [(MultInv\ p\ j) * j * k =$
 $(MultInv\ p\ j) * a] (mod\ p)$
 $\langle proof \rangle$

lemma *aux---4*: $[| 2 < p; \ zprime\ p; \ \sim([j = 0](mod\ p));$
 $[(MultInv\ p\ j) * j * k = (MultInv\ p\ j) * a] (mod\ p) |]$
 $==> [k = a * (MultInv\ p\ j)] (mod\ p)$
 $\langle proof \rangle$

lemma *MultInv-zcong-prop2*: $[| \ 2 < p; \ zprime\ p; \ \sim([k = 0](mod\ p));$
 $\sim([j = 0](mod\ p)); \ [j = a * MultInv\ p\ k] (mod\ p) |] ==>$
 $[k = a * MultInv\ p\ j] (mod\ p)$
 $\langle proof \rangle$

lemma *MultInv-zcong-prop3*: $[| \ 2 < p; \ zprime\ p; \ \sim([a = 0](mod\ p));$
 $\sim([k = 0](mod\ p)); \ \sim([j = 0](mod\ p));$

```

    [a * MultInv p j = a * MultInv p k] (mod p) ==>
    [j = k] (mod p)
  <proof>

end

```

12 Residue Sets

theory *Residues* **imports** *Int2* **begin**

Define the residue of a set, the standard residue, quadratic residues, and prove some basic properties.

definition

```

  ResSet      :: int => int set => bool where
  ResSet m X = (∀ y1 y2. (y1 ∈ X & y2 ∈ X & [y1 = y2] (mod m) --> y1 =
y2))

```

definition

```

  StandardRes :: int => int => int where
  StandardRes m x = x mod m

```

definition

```

  QuadRes     :: int => int => bool where
  QuadRes m x = (∃ y. ([y ^ 2] (mod m)) = x)

```

definition

```

  Legendre    :: int => int => int where
  Legendre a p = (if ([a = 0] (mod p)) then 0
                    else if (QuadRes p a) then 1
                    else -1)

```

definition

```

  SR          :: int => int set where
  SR p = {x. (0 ≤ x) & (x < p)}

```

definition

```

  SRStar      :: int => int set where
  SRStar p = {x. (0 < x) & (x < p)}

```

12.1 Some useful properties of StandardRes

lemma *StandardRes-prop1*: $[x = \text{StandardRes } m \ x] \text{ (mod } m)$
 <proof>

lemma *StandardRes-prop2*: $0 < m ==> (\text{StandardRes } m \ x1 = \text{StandardRes } m \ x2)$
 $= ([x1 = x2] \text{ (mod } m))$

$\langle \text{proof} \rangle$

lemma *StandardRes-prop3*: $(\sim[x = 0] \text{ (mod } p)) = (\sim(\text{StandardRes } p \ x = 0))$
 $\langle \text{proof} \rangle$

lemma *StandardRes-prop4*: $2 < m$
 $\implies [\text{StandardRes } m \ x * \text{StandardRes } m \ y = (x * y)] \text{ (mod } m)$
 $\langle \text{proof} \rangle$

lemma *StandardRes-lbound*: $0 < p \implies 0 \leq \text{StandardRes } p \ x$
 $\langle \text{proof} \rangle$

lemma *StandardRes-ubound*: $0 < p \implies \text{StandardRes } p \ x < p$
 $\langle \text{proof} \rangle$

lemma *StandardRes-eq-zcong*:
 $(\text{StandardRes } m \ x = 0) = ([x = 0] \text{ (mod } m))$
 $\langle \text{proof} \rangle$

12.2 Relations between StandardRes, SRStar, and SR

lemma *SRStar-SR-prop*: $x \in \text{SRStar } p \implies x \in \text{SR } p$
 $\langle \text{proof} \rangle$

lemma *StandardRes-SR-prop*: $x \in \text{SR } p \implies \text{StandardRes } p \ x = x$
 $\langle \text{proof} \rangle$

lemma *StandardRes-SRStar-prop1*: $2 < p \implies (\text{StandardRes } p \ x \in \text{SRStar } p)$
 $= (\sim[x = 0] \text{ (mod } p))$
 $\langle \text{proof} \rangle$

lemma *StandardRes-SRStar-prop1a*: $x \in \text{SRStar } p \implies \sim([x = 0] \text{ (mod } p))$
 $\langle \text{proof} \rangle$

lemma *StandardRes-SRStar-prop2*: $[\ 2 < p; \text{zprime } p; x \in \text{SRStar } p \]$
 $\implies \text{StandardRes } p \ (\text{MultInv } p \ x) \in \text{SRStar } p$
 $\langle \text{proof} \rangle$

lemma *StandardRes-SRStar-prop3*: $x \in \text{SRStar } p \implies \text{StandardRes } p \ x = x$
 $\langle \text{proof} \rangle$

lemma *StandardRes-SRStar-prop4*: $[\ \text{zprime } p; 2 < p; x \in \text{SRStar } p \]$
 $\implies \text{StandardRes } p \ x \in \text{SRStar } p$
 $\langle \text{proof} \rangle$

lemma *SRStar-mult-prop1*: $[\ \text{zprime } p; 2 < p; x \in \text{SRStar } p; y \in \text{SRStar } p \]$
 $\implies (\text{StandardRes } p \ (x * y)) \in \text{SRStar } p$
 $\langle \text{proof} \rangle$

lemma *SRStar-mult-prop2*: $[| \text{zprime } p; 2 < p; \sim([a = 0](\text{mod } p));$
 $x \in \text{SRStar } p \ |]$
 $\implies \text{StandardRes } p \ (a * \text{MultInv } p \ x) \in \text{SRStar } p$
 $\langle \text{proof} \rangle$

lemma *SRStar-card*: $2 < p \implies \text{int}(\text{card}(\text{SRStar } p)) = p - 1$
 $\langle \text{proof} \rangle$

lemma *SRStar-finite*: $2 < p \implies \text{finite}(\text{SRStar } p)$
 $\langle \text{proof} \rangle$

12.3 Properties relating ResSets with StandardRes

lemma *aux*: $x \text{ mod } m = y \text{ mod } m \implies [x = y] (\text{mod } m)$
 $\langle \text{proof} \rangle$

lemma *StandardRes-inj-on-ResSet*: $\text{ResSet } m \ X \implies (\text{inj-on } (\text{StandardRes } m) \ X)$
 $\langle \text{proof} \rangle$

lemma *StandardRes-Sum*: $[| \text{finite } X; 0 < m \ |]$
 $\implies [\text{setsum } f \ X = \text{setsum } (\text{StandardRes } m \ o \ f) \ X](\text{mod } m)$
 $\langle \text{proof} \rangle$

lemma *SR-pos*: $0 < m \implies (\text{StandardRes } m \text{ ' } X) \subseteq \{x. 0 \leq x \ \& \ x < m\}$
 $\langle \text{proof} \rangle$

lemma *ResSet-finite*: $0 < m \implies \text{ResSet } m \ X \implies \text{finite } X$
 $\langle \text{proof} \rangle$

lemma *mod-mod-is-mod*: $[x = x \text{ mod } m](\text{mod } m)$
 $\langle \text{proof} \rangle$

lemma *StandardRes-prod*: $[| \text{finite } X; 0 < m \ |]$
 $\implies [\text{setprod } f \ X = \text{setprod } (\text{StandardRes } m \ o \ f) \ X](\text{mod } m)$
 $\langle \text{proof} \rangle$

lemma *ResSet-image*:
 $[| 0 < m; \text{ResSet } m \ A; \forall x \in A. \forall y \in A. ([f \ x = f \ y](\text{mod } m) \implies x = y) \ |]$
 \implies
 $\text{ResSet } m \ (f \text{ ' } A)$
 $\langle \text{proof} \rangle$

12.4 Property for SRStar

lemma *ResSet-SRStar-prop*: $\text{ResSet } p \ (\text{SRStar } p)$
 $\langle \text{proof} \rangle$

end

13 Parity: Even and Odd Integers

theory *EvenOdd* **imports** *Int2* **begin**

definition

zOdd :: *int set* **where**
zOdd = {*x*. $\exists k. x = 2 * k + 1$ }

definition

zEven :: *int set* **where**
zEven = {*x*. $\exists k. x = 2 * k$ }

13.1 Some useful properties about even and odd

lemma *zOddI* [*intro?*]: $x = 2 * k + 1 \implies x \in zOdd$
and *zOddE* [*elim?*]: $x \in zOdd \implies (!k. x = 2 * k + 1 \implies C) \implies C$
 <proof>

lemma *zEvenI* [*intro?*]: $x = 2 * k \implies x \in zEven$
and *zEvenE* [*elim?*]: $x \in zEven \implies (!k. x = 2 * k \implies C) \implies C$
 <proof>

lemma *one-not-even*: $\sim(1 \in zEven)$
 <proof>

lemma *even-odd-conj*: $\sim(x \in zOdd \ \& \ x \in zEven)$
 <proof>

lemma *even-odd-disj*: $(x \in zOdd \mid x \in zEven)$
 <proof>

lemma *not-odd-impl-even*: $\sim(x \in zOdd) \implies x \in zEven$
 <proof>

lemma *odd-mult-odd-prop*: $(x*y):zOdd \implies x \in zOdd$
 <proof>

lemma *odd-minus-one-even*: $x \in zOdd \implies (x - 1):zEven$
 <proof>

lemma *even-div-2-prop1*: $x \in zEven \implies (x \bmod 2) = 0$
 <proof>

lemma *even-div-2-prop2*: $x \in zEven \implies (2 * (x \div 2)) = x$
 <proof>

lemma *even-plus-even*: $[| x \in zEven; y \in zEven |] \implies x + y \in zEven$
 <proof>

lemma *even-times-either*: $x \in zEven \implies x * y \in zEven$

$\langle \text{proof} \rangle$

lemma *even-minus-even*: $[[x \in \text{zEven}; y \in \text{zEven}]] \implies x - y \in \text{zEven}$
 $\langle \text{proof} \rangle$

lemma *odd-minus-odd*: $[[x \in \text{zOdd}; y \in \text{zOdd}]] \implies x - y \in \text{zEven}$
 $\langle \text{proof} \rangle$

lemma *even-minus-odd*: $[[x \in \text{zEven}; y \in \text{zOdd}]] \implies x - y \in \text{zOdd}$
 $\langle \text{proof} \rangle$

lemma *odd-minus-even*: $[[x \in \text{zOdd}; y \in \text{zEven}]] \implies x - y \in \text{zOdd}$
 $\langle \text{proof} \rangle$

lemma *odd-times-odd*: $[[x \in \text{zOdd}; y \in \text{zOdd}]] \implies x * y \in \text{zOdd}$
 $\langle \text{proof} \rangle$

lemma *odd-iff-not-even*: $(x \in \text{zOdd}) = (\sim (x \in \text{zEven}))$
 $\langle \text{proof} \rangle$

lemma *even-product*: $x * y \in \text{zEven} \implies x \in \text{zEven} \mid y \in \text{zEven}$
 $\langle \text{proof} \rangle$

lemma *even-diff*: $x - y \in \text{zEven} = ((x \in \text{zEven}) = (y \in \text{zEven}))$
 $\langle \text{proof} \rangle$

lemma *neg-one-even-power*: $[[x \in \text{zEven}; 0 \leq x]] \implies (-1::\text{int})^{\text{nat } x} = 1$
 $\langle \text{proof} \rangle$

lemma *neg-one-odd-power*: $[[x \in \text{zOdd}; 0 \leq x]] \implies (-1::\text{int})^{\text{nat } x} = -1$
 $\langle \text{proof} \rangle$

lemma *neg-one-power-parity*: $[[0 \leq x; 0 \leq y; (x \in \text{zEven}) = (y \in \text{zEven})]] \implies$
 $(-1::\text{int})^{\text{nat } x} = (-1::\text{int})^{\text{nat } y}$
 $\langle \text{proof} \rangle$

lemma *one-not-neg-one-mod-m*: $2 < m \implies \sim([1 = -1] \text{ (mod } m))$
 $\langle \text{proof} \rangle$

lemma *even-div-2-l*: $[[y \in \text{zEven}; x < y]] \implies x \text{ div } 2 < y \text{ div } 2$
 $\langle \text{proof} \rangle$

lemma *even-sum-div-2*: $[[x \in \text{zEven}; y \in \text{zEven}]] \implies (x + y) \text{ div } 2 = x \text{ div } 2$
 $+ y \text{ div } 2$
 $\langle \text{proof} \rangle$

lemma *even-prod-div-2*: $[[x \in \text{zEven}]] \implies (x * y) \text{ div } 2 = (x \text{ div } 2) * y$
 $\langle \text{proof} \rangle$

lemma *zprime-zOdd-eq-grt-2*: $zprime\ p \implies (p \in zOdd) = (2 < p)$
 $\langle proof \rangle$

lemma *neg-one-special*: $finite\ A \implies$
 $((-1 :: int) ^ card\ A) * (-1 ^ card\ A) = 1$
 $\langle proof \rangle$

lemma *neg-one-power*: $(-1 :: int) ^ n = 1 \mid (-1 :: int) ^ n = -1$
 $\langle proof \rangle$

lemma *neg-one-power-eq-mod-m*: $[\mid 2 < m; [(-1 :: int) ^ j = (-1 :: int) ^ k] \ (mod\ m)$
 $\mid]$
 $\implies ((-1 :: int) ^ j = (-1 :: int) ^ k)$
 $\langle proof \rangle$

end

14 Euler's criterion

theory *Euler* **imports** *Residues EvenOdd* **begin**

definition

MultiInvPair :: $int \Rightarrow int \Rightarrow int \Rightarrow int\ set$ **where**
 $MultiInvPair\ a\ p\ j = \{StandardRes\ p\ j, StandardRes\ p\ (a * (MultiInv\ p\ j))\}$

definition

SetS :: $int \Rightarrow int \Rightarrow int\ set\ set$ **where**
 $SetS\ a\ p = (MultiInvPair\ a\ p\ 'SRStar\ p)$

14.1 Property for MultiInvPair

lemma *MultiInvPair-prop1a*:

$[\mid zprime\ p; 2 < p; \sim([a = 0](mod\ p));$
 $X \in (SetS\ a\ p); Y \in (SetS\ a\ p);$
 $\sim((X \cap Y) = \{\}) \mid] \implies X = Y$
 $\langle proof \rangle$

lemma *MultiInvPair-prop1b*:

$[\mid zprime\ p; 2 < p; \sim([a = 0](mod\ p));$
 $X \in (SetS\ a\ p); Y \in (SetS\ a\ p);$
 $X \neq Y \mid] \implies X \cap Y = \{\}$
 $\langle proof \rangle$

lemma *MultInvPair-prop1c*: $\llbracket \text{zprime } p; 2 < p; \sim([a = 0](\text{mod } p)) \rrbracket \implies$
 $\forall X \in \text{SetS } a \ p. \forall Y \in \text{SetS } a \ p. X \neq Y \dashv\dashv X \cap Y = \{\}$
 $\langle \text{proof} \rangle$

lemma *MultInvPair-prop2*: $\llbracket \text{zprime } p; 2 < p; \sim([a = 0](\text{mod } p)) \rrbracket \implies$
 $\text{Union } (\text{SetS } a \ p) = \text{SRStar } p$
 $\langle \text{proof} \rangle$

lemma *MultInvPair-distinct*: $\llbracket \text{zprime } p; 2 < p; \sim([a = 0](\text{mod } p));$
 $\sim([j = 0](\text{mod } p));$
 $\sim(\text{QuadRes } p \ a) \rrbracket \implies$
 $\sim([j = a * \text{MultInv } p \ j](\text{mod } p))$
 $\langle \text{proof} \rangle$

lemma *MultInvPair-card-two*: $\llbracket \text{zprime } p; 2 < p; \sim([a = 0](\text{mod } p));$
 $\sim(\text{QuadRes } p \ a); \sim([j = 0](\text{mod } p)) \rrbracket \implies$
 $\text{card } (\text{MultInvPair } a \ p \ j) = 2$
 $\langle \text{proof} \rangle$

14.2 Properties of SetS

lemma *SetS-finite*: $2 < p \implies \text{finite } (\text{SetS } a \ p)$
 $\langle \text{proof} \rangle$

lemma *SetS-elems-finite*: $\forall X \in \text{SetS } a \ p. \text{finite } X$
 $\langle \text{proof} \rangle$

lemma *SetS-elems-card*: $\llbracket \text{zprime } p; 2 < p; \sim([a = 0](\text{mod } p));$
 $\sim(\text{QuadRes } p \ a) \rrbracket \implies$
 $\forall X \in \text{SetS } a \ p. \text{card } X = 2$
 $\langle \text{proof} \rangle$

lemma *Union-SetS-finite*: $2 < p \implies \text{finite } (\text{Union } (\text{SetS } a \ p))$
 $\langle \text{proof} \rangle$

lemma *card-setsum-aux*: $\llbracket \text{finite } S; \forall X \in S. \text{finite } (X::\text{int set});$
 $\forall X \in S. \text{card } X = n \rrbracket \implies \text{setsum } \text{card } S = \text{setsum } (\%x. n) \ S$
 $\langle \text{proof} \rangle$

lemma *SetS-card*: $\llbracket \text{zprime } p; 2 < p; \sim([a = 0](\text{mod } p)); \sim(\text{QuadRes } p \ a) \rrbracket$
 \implies
 $\text{int}(\text{card}(\text{SetS } a \ p)) = (p - 1) \text{ div } 2$
 $\langle \text{proof} \rangle$

lemma *SetS-setprod-prop*: $\llbracket \text{zprime } p; 2 < p; \sim([a = 0](\text{mod } p));$
 $\sim(\text{QuadRes } p \ a); x \in (\text{SetS } a \ p) \rrbracket \implies$
 $\prod x = a \ (\text{mod } p)$
 $\langle \text{proof} \rangle$

lemma *aux1*: $[[0 < x; (x::int) < a; x \neq (a - 1)]] \implies x < a - 1$
 $\langle proof \rangle$

lemma *aux2*: $[[(a::int) < c; b < c]] \implies (a \leq b \mid b \leq a)$
 $\langle proof \rangle$

lemma *SRStar-d2set-prop*: $2 < p \implies (SRStar\ p) = \{1\} \cup (d2set\ (p - 1))$
 $\langle proof \rangle$

lemma *Union-SetS-setprod-prop1*: $[[zprime\ p; 2 < p; \sim([a = 0] \ (mod\ p)); \sim(QuadRes\ p\ a)]] \implies$

$$[[\prod (Union\ (SetS\ a\ p)) = a \wedge nat\ ((p - 1) \div 2)]] \ (mod\ p)$$

 $\langle proof \rangle$

lemma *Union-SetS-setprod-prop2*: $[[zprime\ p; 2 < p; \sim([a = 0] \ (mod\ p))]] \implies$

$$\prod (Union\ (SetS\ a\ p)) = zfact\ (p - 1)$$

 $\langle proof \rangle$

lemma *zfact-prop*: $[[zprime\ p; 2 < p; \sim([a = 0] \ (mod\ p)); \sim(QuadRes\ p\ a)]] \implies$

$$[zfact\ (p - 1) = a \wedge nat\ ((p - 1) \div 2)] \ (mod\ p)$$

 $\langle proof \rangle$

Prove the first part of Euler's Criterion:

lemma *Euler-part1*: $[[2 < p; zprime\ p; \sim([x = 0] \ (mod\ p)); \sim(QuadRes\ p\ x)]] \implies$

$$[x \wedge nat\ (((p) - 1) \div 2) = -1] \ (mod\ p)$$

 $\langle proof \rangle$

Prove another part of Euler Criterion:

lemma *aux-1*: $0 < p \implies (a::int) \wedge nat\ (p) = a * a \wedge (nat\ (p) - 1)$
 $\langle proof \rangle$

lemma *aux-2*: $[[2::int) < p; p \in zOdd]] \implies 0 < ((p - 1) \div 2)$
 $\langle proof \rangle$

lemma *Euler-part2*:
 $[[2 < p; zprime\ p; [a = 0] \ (mod\ p)]] \implies [0 = a \wedge nat\ ((p - 1) \div 2)] \ (mod\ p)$
 $\langle proof \rangle$

Prove the final part of Euler's Criterion:

lemma *aux--1*: $[[\sim([x = 0] \ (mod\ p)); [y \wedge 2 = x] \ (mod\ p)]] \implies \sim(p \mid y)$
 $\langle proof \rangle$

lemma *aux-2*: $2 * \text{nat}((p - 1) \text{ div } 2) = \text{nat}(2 * ((p - 1) \text{ div } 2))$
 ⟨proof⟩

lemma *Euler-part3*: $[[2 < p; \text{zprime } p; \sim([x = 0](\text{mod } p)); \text{QuadRes } p \ x]] ==>$
 $[x^{\text{nat}(((p) - 1) \text{ div } 2)} = 1](\text{mod } p)$
 ⟨proof⟩

Finally show Euler's Criterion:

theorem *Euler-Criterion*: $[[2 < p; \text{zprime } p]] ==> [(\text{Legendre } a \ p) =$
 $a^{\text{nat}(((p) - 1) \text{ div } 2)}](\text{mod } p)$
 ⟨proof⟩

end

15 Gauss' Lemma

theory *Gauss* **imports** *Euler* **begin**

locale *GAUSS* =

fixes $p :: \text{int}$

fixes $a :: \text{int}$

assumes $p\text{-prime}$: $\text{zprime } p$

assumes $p\text{-g-2}$: $2 < p$

assumes $p\text{-a-relprime}$: $\sim[a = 0](\text{mod } p)$

assumes $a\text{-nonzero}$: $0 < a$

begin

definition

$A :: \text{int set}$ **where**

$A = \{(x::\text{int}). 0 < x \ \& \ x \leq ((p - 1) \text{ div } 2)\}$

definition

$B :: \text{int set}$ **where**

$B = (\%x. x * a) \text{ ` } A$

definition

$C :: \text{int set}$ **where**

$C = \text{StandardRes } p \text{ ` } B$

definition

$D :: \text{int set}$ **where**

$D = C \cap \{x. x \leq ((p - 1) \text{ div } 2)\}$

definition

$E :: \text{int set}$ **where**

$$E = C \cap \{x. ((p - 1) \text{ div } 2) < x\}$$

definition

$F :: \text{int set}$ **where**
 $F = (\%x. (p - x)) \text{ ' } E$

15.1 Basic properties of p

lemma *p-odd*: $p \in \text{zOdd}$
 $\langle \text{proof} \rangle$

lemma *p-g-0*: $0 < p$
 $\langle \text{proof} \rangle$

lemma *int-nat*: $\text{int } (\text{nat } ((p - 1) \text{ div } 2)) = (p - 1) \text{ div } 2$
 $\langle \text{proof} \rangle$

lemma *p-minus-one-l*: $(p - 1) \text{ div } 2 < p$
 $\langle \text{proof} \rangle$

lemma *p-eq*: $p = (2 * (p - 1) \text{ div } 2) + 1$
 $\langle \text{proof} \rangle$

lemma (**in** $-$) *zodd-imp-zdiv-eq*: $x \in \text{zOdd} ==> 2 * (x - 1) \text{ div } 2 = 2 * ((x - 1) \text{ div } 2)$
 $\langle \text{proof} \rangle$

lemma *p-eq2*: $p = (2 * ((p - 1) \text{ div } 2)) + 1$
 $\langle \text{proof} \rangle$

15.2 Basic Properties of the Gauss Sets

lemma *finite-A*: $\text{finite } (A)$
 $\langle \text{proof} \rangle$

lemma *finite-B*: $\text{finite } (B)$
 $\langle \text{proof} \rangle$

lemma *finite-C*: $\text{finite } (C)$
 $\langle \text{proof} \rangle$

lemma *finite-D*: $\text{finite } (D)$
 $\langle \text{proof} \rangle$

lemma *finite-E*: $\text{finite } (E)$
 $\langle \text{proof} \rangle$

lemma *finite-F*: $\text{finite } (F)$

$\langle proof \rangle$

lemma *C-eq*: $C = D \cup E$
 $\langle proof \rangle$

lemma *A-card-eq*: $card\ A = nat\ ((p - 1) \div 2)$
 $\langle proof \rangle$

lemma *inj-on-xa-A*: $inj\text{-}on\ (\%x. x * a)\ A$
 $\langle proof \rangle$

lemma *A-res*: $ResSet\ p\ A$
 $\langle proof \rangle$

lemma *B-res*: $ResSet\ p\ B$
 $\langle proof \rangle$

lemma *SR-B-inj*: $inj\text{-}on\ (StandardRes\ p)\ B$
 $\langle proof \rangle$

lemma *inj-on-pminusx-E*: $inj\text{-}on\ (\%x. p - x)\ E$
 $\langle proof \rangle$

lemma *A-ncong-p*: $x \in A ==> \sim[x = 0](mod\ p)$
 $\langle proof \rangle$

lemma *A-greater-zero*: $x \in A ==> 0 < x$
 $\langle proof \rangle$

lemma *B-ncong-p*: $x \in B ==> \sim[x = 0](mod\ p)$
 $\langle proof \rangle$

lemma *B-greater-zero*: $x \in B ==> 0 < x$
 $\langle proof \rangle$

lemma *C-ncong-p*: $x \in C ==> \sim[x = 0](mod\ p)$
 $\langle proof \rangle$

lemma *C-greater-zero*: $y \in C ==> 0 < y$
 $\langle proof \rangle$

lemma *D-ncong-p*: $x \in D ==> \sim[x = 0](mod\ p)$
 $\langle proof \rangle$

lemma *E-ncong-p*: $x \in E ==> \sim[x = 0](mod\ p)$
 $\langle proof \rangle$

lemma *F-ncong-p*: $x \in F ==> \sim[x = 0](mod\ p)$
 $\langle proof \rangle$

lemma *F-subset*: $F \subseteq \{x. 0 < x \ \& \ x \leq ((p - 1) \text{ div } 2)\}$
 $\langle \text{proof} \rangle$

lemma *D-subset*: $D \subseteq \{x. 0 < x \ \& \ x \leq ((p - 1) \text{ div } 2)\}$
 $\langle \text{proof} \rangle$

lemma *F-eq*: $F = \{x. \exists y \in A. (x = p - (\text{StandardRes } p \ (y * a)) \ \& \ (p - 1) \text{ div } 2 < \text{StandardRes } p \ (y * a))\}$
 $\langle \text{proof} \rangle$

lemma *D-eq*: $D = \{x. \exists y \in A. (x = \text{StandardRes } p \ (y * a) \ \& \ \text{StandardRes } p \ (y * a) \leq (p - 1) \text{ div } 2)\}$
 $\langle \text{proof} \rangle$

lemma *D-leq*: $x \in D \implies x \leq (p - 1) \text{ div } 2$
 $\langle \text{proof} \rangle$

lemma *F-ge*: $x \in F \implies x \leq (p - 1) \text{ div } 2$
 $\langle \text{proof} \rangle$

lemma *all-A-relprime*: $\forall x \in A. \text{zgcd}(x, p) = 1$
 $\langle \text{proof} \rangle$

lemma *A-prod-relprime*: $\text{zgcd}((\text{setprod id } A), p) = 1$
 $\langle \text{proof} \rangle$

15.3 Relationships Between Gauss Sets

lemma *B-card-eq-A*: $\text{card } B = \text{card } A$
 $\langle \text{proof} \rangle$

lemma *B-card-eq*: $\text{card } B = \text{nat } ((p - 1) \text{ div } 2)$
 $\langle \text{proof} \rangle$

lemma *F-card-eq-E*: $\text{card } F = \text{card } E$
 $\langle \text{proof} \rangle$

lemma *C-card-eq-B*: $\text{card } C = \text{card } B$
 $\langle \text{proof} \rangle$

lemma *D-E-disj*: $D \cap E = \{\}$
 $\langle \text{proof} \rangle$

lemma *C-card-eq-D-plus-E*: $\text{card } C = \text{card } D + \text{card } E$
 $\langle \text{proof} \rangle$

lemma *C-prod-eq-D-times-E*: $\text{setprod id } E * \text{setprod id } D = \text{setprod id } C$
 $\langle \text{proof} \rangle$

lemma *C-B-zcong-prod*: $[setprod\ id\ C = setprod\ id\ B] \pmod p$
 $\langle proof \rangle$

lemma *F-Un-D-subset*: $(F \cup D) \subseteq A$
 $\langle proof \rangle$

lemma *F-D-disj*: $(F \cap D) = \{\}$
 $\langle proof \rangle$

lemma *F-Un-D-card*: $card\ (F \cup D) = nat\ ((p - 1) \div 2)$
 $\langle proof \rangle$

lemma *F-Un-D-eq-A*: $F \cup D = A$
 $\langle proof \rangle$

lemma *prod-D-F-eq-prod-A*:
 $(setprod\ id\ D) * (setprod\ id\ F) = setprod\ id\ A$
 $\langle proof \rangle$

lemma *prod-F-zcong*:
 $[setprod\ id\ F = ((-1) ^ (card\ E)) * (setprod\ id\ E)] \pmod p$
 $\langle proof \rangle$

15.4 Gauss' Lemma

lemma *aux*: $setprod\ id\ A * -1 ^ card\ E * a ^ card\ A * -1 ^ card\ E = setprod\ id\ A * a ^ card\ A$
 $\langle proof \rangle$

theorem *pre-gauss-lemma*:
 $[a ^ nat((p - 1) \div 2) = (-1) ^ (card\ E)] \pmod p$
 $\langle proof \rangle$

theorem *gauss-lemma*: $(Legendre\ a\ p) = (-1) ^ (card\ E)$
 $\langle proof \rangle$

end

end

16 The law of Quadratic reciprocity

theory *Quadratic-Reciprocity*
imports *Gauss*
begin

Lemmas leading up to the proof of theorem 3.3 in Niven and Zuckerman's

presentation.

context *GAUSS*

begin

lemma *QRLemma1*: $a * \text{setsum id } A =$
 $p * \text{setsum } (\%x. ((x * a) \text{ div } p)) A + \text{setsum id } D + \text{setsum id } E$
 $\langle \text{proof} \rangle$

lemma *QRLemma2*: $\text{setsum id } A = p * \text{int } (\text{card } E) - \text{setsum id } E +$
 $\text{setsum id } D$
 $\langle \text{proof} \rangle$

lemma *QRLemma3*: $(a - 1) * \text{setsum id } A =$
 $p * (\text{setsum } (\%x. ((x * a) \text{ div } p)) A - \text{int}(\text{card } E)) + 2 * \text{setsum id } E$
 $\langle \text{proof} \rangle$

lemma *QRLemma4*: $a \in \text{zOdd} ==>$
 $(\text{setsum } (\%x. ((x * a) \text{ div } p)) A \in \text{zEven}) = (\text{int}(\text{card } E) \in \text{zEven})$
 $\langle \text{proof} \rangle$

lemma *QRLemma5*: $a \in \text{zOdd} ==>$
 $(-1::\text{int})^{\text{card } E} = (-1::\text{int})^{\text{nat}(\text{setsum } (\%x. ((x * a) \text{ div } p)) A)}$
 $\langle \text{proof} \rangle$

end

lemma *MainQRLemma*: $[| a \in \text{zOdd}; 0 < a; \sim([a = 0] \text{ mod } p); \text{zprime } p; 2 <$
 $p;$
 $A = \{x. 0 < x \ \& \ x \leq (p - 1) \text{ div } 2\} |] ==>$
 $(\text{Legendre } a \ p) = (-1::\text{int})^{\text{nat}(\text{setsum } (\%x. ((x * a) \text{ div } p)) A)}$
 $\langle \text{proof} \rangle$

16.1 Stuff about S, S1 and S2

locale *QRTEMP* =

fixes $p \quad :: \text{int}$

fixes $q \quad :: \text{int}$

assumes $p\text{-prime}$: $\text{zprime } p$

assumes $p\text{-g-2}$: $2 < p$

assumes $q\text{-prime}$: $\text{zprime } q$

assumes $q\text{-g-2}$: $2 < q$

assumes $p\text{-neq-}q$: $p \neq q$

begin

definition

$P\text{-set} :: \text{int set}$ **where**

$P\text{-set} = \{x. 0 < x \ \& \ x \leq ((p - 1) \text{ div } 2)\}$

definition

$Q\text{-set} :: \text{int set}$ **where**
 $Q\text{-set} = \{x. 0 < x \ \& \ x \leq ((q - 1) \text{ div } 2)\}$

definition

$S :: (\text{int} * \text{int}) \text{ set}$ **where**
 $S = P\text{-set} <*> Q\text{-set}$

definition

$S1 :: (\text{int} * \text{int}) \text{ set}$ **where**
 $S1 = \{(x, y). (x, y):S \ \& \ ((p * y) < (q * x))\}$

definition

$S2 :: (\text{int} * \text{int}) \text{ set}$ **where**
 $S2 = \{(x, y). (x, y):S \ \& \ ((q * x) < (p * y))\}$

definition

$f1 :: \text{int} \Rightarrow (\text{int} * \text{int}) \text{ set}$ **where**
 $f1 \ j = \{(j1, y). (j1, y):S \ \& \ j1 = j \ \& \ (y \leq (q * j) \text{ div } p)\}$

definition

$f2 :: \text{int} \Rightarrow (\text{int} * \text{int}) \text{ set}$ **where**
 $f2 \ j = \{(x, j1). (x, j1):S \ \& \ j1 = j \ \& \ (x \leq (p * j) \text{ div } q)\}$

lemma $p\text{-fact}$: $0 < (p - 1) \text{ div } 2$
 $\langle \text{proof} \rangle$

lemma $q\text{-fact}$: $0 < (q - 1) \text{ div } 2$
 $\langle \text{proof} \rangle$

lemma $pb\text{-neq}\text{-}qa$: $[1 \leq b; b \leq (q - 1) \text{ div } 2] \Rightarrow$
 $(p * b \neq q * a)$
 $\langle \text{proof} \rangle$

lemma $P\text{-set}\text{-finite}$: $\text{finite } (P\text{-set})$
 $\langle \text{proof} \rangle$

lemma $Q\text{-set}\text{-finite}$: $\text{finite } (Q\text{-set})$
 $\langle \text{proof} \rangle$

lemma $S\text{-finite}$: $\text{finite } S$
 $\langle \text{proof} \rangle$

lemma $S1\text{-finite}$: $\text{finite } S1$
 $\langle \text{proof} \rangle$

lemma $S2\text{-finite}$: $\text{finite } S2$
 $\langle \text{proof} \rangle$

lemma *P-set-card*: $(p - 1) \text{ div } 2 = \text{int } (\text{card } (P\text{-set}))$
 $\langle \text{proof} \rangle$

lemma *Q-set-card*: $(q - 1) \text{ div } 2 = \text{int } (\text{card } (Q\text{-set}))$
 $\langle \text{proof} \rangle$

lemma *S-card*: $((p - 1) \text{ div } 2) * ((q - 1) \text{ div } 2) = \text{int } (\text{card}(S))$
 $\langle \text{proof} \rangle$

lemma *S1-Int-S2-prop*: $S1 \cap S2 = \{\}$
 $\langle \text{proof} \rangle$

lemma *S1-Union-S2-prop*: $S = S1 \cup S2$
 $\langle \text{proof} \rangle$

lemma *card-sum-S1-S2*: $((p - 1) \text{ div } 2) * ((q - 1) \text{ div } 2) =$
 $\text{int}(\text{card}(S1)) + \text{int}(\text{card}(S2))$
 $\langle \text{proof} \rangle$

lemma *aux1a*: $\llbracket 0 < a; a \leq (p - 1) \text{ div } 2;$
 $0 < b; b \leq (q - 1) \text{ div } 2 \rrbracket ==>$
 $(p * b < q * a) = (b \leq q * a \text{ div } p)$
 $\langle \text{proof} \rangle$

lemma *aux1b*: $\llbracket 0 < a; a \leq (p - 1) \text{ div } 2;$
 $0 < b; b \leq (q - 1) \text{ div } 2 \rrbracket ==>$
 $(q * a < p * b) = (a \leq p * b \text{ div } q)$
 $\langle \text{proof} \rangle$

lemma (*in -*) *aux2*: $\llbracket \text{zprime } p; \text{zprime } q; 2 < p; 2 < q \rrbracket ==>$
 $(q * ((p - 1) \text{ div } 2)) \text{ div } p \leq (q - 1) \text{ div } 2$
 $\langle \text{proof} \rangle$

lemma *aux3a*: $\forall j \in P\text{-set}. \text{int } (\text{card } (f1 \ j)) = (q * j) \text{ div } p$
 $\langle \text{proof} \rangle$

lemma *aux3b*: $\forall j \in Q\text{-set}. \text{int } (\text{card } (f2 \ j)) = (p * j) \text{ div } q$
 $\langle \text{proof} \rangle$

lemma *S1-card*: $\text{int } (\text{card}(S1)) = \text{setsum } (\%j. (q * j) \text{ div } p) \ P\text{-set}$
 $\langle \text{proof} \rangle$

lemma *S2-card*: $\text{int } (\text{card}(S2)) = \text{setsum } (\%j. (p * j) \text{ div } q) \ Q\text{-set}$
 $\langle \text{proof} \rangle$

lemma *S1-carda*: $\text{int } (\text{card}(S1)) =$
 $\text{setsum } (\%j. (j * q) \text{ div } p) \ P\text{-set}$
 $\langle \text{proof} \rangle$

lemma *S2-carda*: $\text{int } (\text{card}(S2)) =$
 $\text{setsum } (\%j. (j * p) \text{ div } q) \text{ } Q\text{-set}$
 $\langle \text{proof} \rangle$

lemma *pq-sum-prop*: $(\text{setsum } (\%j. (j * p) \text{ div } q) \text{ } Q\text{-set}) +$
 $(\text{setsum } (\%j. (j * q) \text{ div } p) \text{ } P\text{-set}) = ((p - 1) \text{ div } 2) * ((q - 1) \text{ div } 2)$
 $\langle \text{proof} \rangle$

lemma (**in** $-$) *pq-prime-neg*: $[[\text{zprime } p; \text{zprime } q; p \neq q]] ==> (\sim [p = 0] \text{ (mod } q))$
 $\langle \text{proof} \rangle$

lemma *QR-short*: $(\text{Legendre } p \text{ } q) * (\text{Legendre } q \text{ } p) =$
 $(-1::\text{int})^{\text{nat}(((p - 1) \text{ div } 2) * ((q - 1) \text{ div } 2))}$
 $\langle \text{proof} \rangle$

end

theorem *Quadratic-Reciprocity*:
 $[[p \in \text{zOdd}; \text{zprime } p; q \in \text{zOdd}; \text{zprime } q;$
 $p \neq q]]$
 $==> (\text{Legendre } p \text{ } q) * (\text{Legendre } q \text{ } p) =$
 $(-1::\text{int})^{\text{nat}(((p - 1) \text{ div } 2) * ((q - 1) \text{ div } 2))}$
 $\langle \text{proof} \rangle$

end