

Gauss Sums and the Pólya–Vinogradov Inequality

Rodrigo Raya and Manuel Eberl

February 23, 2021

Abstract

This article provides a full formalisation of Chapter 8 of Apostol’s *Introduction to Analytic Number Theory* [1]. Subjects that are covered are:

- periodic arithmetic functions and their finite Fourier series
- (generalised) Ramanujan sums
- Gauss sums and separable characters
- induced moduli and primitive characters
- the Pólya–Vinogradov inequality

Contents

1	Auxiliary material	3
1.1	Various facts	3
1.2	Neutral element of the Dirichlet product	9
1.3	Multiplicative functions	10
2	Periodic arithmetic functions	11
3	Complex roots of unity	18
4	Geometric sums of roots of unity	22
5	Finite Fourier series	23
5.1	Auxiliary facts	23
5.2	Definition and uniqueness	25
5.3	Expansion of an arithmetical function	30
6	Ramanujan sums	34
6.1	Basic sums	34
6.2	Generalised sums	37

7	Gauss sums	51
7.1	Definition and basic properties	51
7.2	Separability	55
7.3	Induced moduli and primitive characters	59
7.4	The conductor of a character	68
7.5	The connection between primitivity and separability	71
8	The Pólya–Vinogradov Inequality	87
8.1	The case of primitive characters	87
8.2	General case	101

1 Auxiliary material

```
theory Gauss-Sums-Auxiliary
imports
  Dirichlet-L.Dirichlet-Characters
  Dirichlet-Series.Moebius-Mu
  Dirichlet-Series.More-Totient
begin
```

1.1 Various facts

lemma *sum-div-reduce*:

```
fixes d :: nat and f :: nat ⇒ complex
assumes d dvd k d > 0
shows (∑ n | n ∈ {1..k} ∧ d dvd n. f n) = (∑ c ∈ {1..k div d}. f (c*d))
by (rule sum.reindex-bij-witness[of - λk. k * d λk. k div d])
  (use assms in ⟨fastforce simp: div-le-mono⟩+)
```

lemma *prod-div-sub*:

```
fixes f :: nat ⇒ complex
assumes finite A B ⊆ A ∀ b ∈ B. f b ≠ 0
shows (∏ i ∈ A - B. f i) = ((∏ i ∈ A. f i) div (∏ i ∈ B. f i))
using assms
```

proof (induction card B arbitrary: B)

case 0

```
then show ?case
  using infinite-super by fastforce
```

next

```
case (Suc n)
then show ?case
```

proof -

```
obtain B' x where decomp: B = B' ∪ {x} ∧ x ∉ B'
  using card-eq-SucD[OF Suc(2)[symmetric]] insert-is-Un by auto
then have B'card: card B' = n using Suc(2)
  using Suc.premis(2) assms(1) finite-subset by fastforce
have prod f (A - B) = prod f ((A - B') - {x})
  by (simp add: decomp, subst Diff-insert, simp)
also have ... = (prod f (A - B')) div f x
  using prod-diff1[of A - B' f x] Suc decomp by auto
also have ... = (prod f A div prod f B') div f x
  using Suc(1)[of B'] Suc(3) B'card decomp
  Suc.premis(2) Suc.premis(3) by force
also have ... = prod f A div (prod f B' * f x) by auto
also have ... = prod f A div prod f B
  using decomp Suc.premis(2) assms(1) finite-subset by fastforce
finally show ?thesis by blast
```

qed

qed

lemma *linear-gcd*:

```

fixes  $a\ b\ c\ d :: \text{nat}$ 
assumes  $a > 0\ b > 0\ c > 0\ d > 0$ 
assumes  $\text{coprime}\ a\ c\ \text{coprime}\ b\ d$ 
shows  $\text{gcd}\ (a*b)\ (c*d) = (\text{gcd}\ a\ d) * (\text{gcd}\ b\ c)$ 
using  $\text{assms}$ 
proof –
  define  $q1 :: \text{nat}$  where  $q1 = a\ \text{div}\ \text{gcd}\ a\ d$ 
  define  $q2 :: \text{nat}$  where  $q2 = c\ \text{div}\ \text{gcd}\ b\ c$ 
  define  $q3 :: \text{nat}$  where  $q3 = b\ \text{div}\ \text{gcd}\ b\ c$ 
  define  $q4 :: \text{nat}$  where  $q4 = d\ \text{div}\ \text{gcd}\ a\ d$ 

  have  $\text{coprime}\ q1\ q2\ \text{coprime}\ q3\ q4$ 
    unfolding  $q1\text{-def}\ q2\text{-def}\ q3\text{-def}\ q4\text{-def}$ 
  proof –
    have  $\text{coprime}\ (a\ \text{div}\ \text{gcd}\ a\ d)\ c$ 
      using  $\langle \text{coprime}\ a\ c \rangle\ \text{coprime-mult-left-iff}[of\ a\ \text{div}\ \text{gcd}\ a\ d\ \text{gcd}\ a\ d\ c]$ 
         $\text{dvd-mult-div-cancel}[OF\ \text{gcd-dvd1},\ of\ a\ b]$  by  $\text{simp}$ 
    then show  $\text{coprime}\ (a\ \text{div}\ \text{gcd}\ a\ d)\ (c\ \text{div}\ \text{gcd}\ b\ c)$ 
      using  $\text{coprime-mult-right-iff}[of\ a\ \text{div}\ \text{gcd}\ a\ d\ \text{gcd}\ b\ c\ c\ \text{div}\ \text{gcd}\ b\ c]$ 
         $\text{dvd-div-mult-self}[OF\ \text{gcd-dvd2}[of\ b\ c]]$  by  $\text{auto}$ 
    have  $\text{coprime}\ (b\ \text{div}\ \text{gcd}\ b\ c)\ d$ 
      using  $\langle \text{coprime}\ b\ d \rangle\ \text{coprime-mult-left-iff}[of\ b\ \text{div}\ \text{gcd}\ b\ c\ \text{gcd}\ b\ c\ d]$ 
         $\text{dvd-mult-div-cancel}[OF\ \text{gcd-dvd1},\ of\ a\ b]$  by  $\text{simp}$ 
    then show  $\text{coprime}\ (b\ \text{div}\ \text{gcd}\ b\ c)\ (d\ \text{div}\ \text{gcd}\ a\ d)$ 
      using  $\text{coprime-mult-right-iff}[of\ b\ \text{div}\ \text{gcd}\ b\ c\ \text{gcd}\ a\ d\ d\ \text{div}\ \text{gcd}\ a\ d]$ 
         $\text{dvd-div-mult-self}[OF\ \text{gcd-dvd2}[of\ b\ c]]$  by  $\text{auto}$ 
  qed
  moreover have  $\text{coprime}\ q1\ q4\ \text{coprime}\ q3\ q2$ 
    unfolding  $q1\text{-def}\ q2\text{-def}\ q3\text{-def}\ q4\text{-def}$ 
    using  $\text{assms}\ \text{div-gcd-coprime}$  by  $\text{blast+}$ 
  ultimately have  $1: \text{coprime}\ (q1*q3)\ (q2*q4)$ 
    by  $\text{simp}$ 
  have  $\text{gcd}\ (a*b)\ (c*d) = (\text{gcd}\ a\ d) * (\text{gcd}\ b\ c) * \text{gcd}\ (q1*q3)\ (q2*q4)$ 
    unfolding  $q1\text{-def}\ q2\text{-def}\ q3\text{-def}\ q4\text{-def}$ 
    by  $(\text{subst}\ \text{gcd-mult-distrib-nat}[of\ \text{gcd}\ a\ d * \text{gcd}\ b\ c],$ 
       $\text{simp}\ \text{add:}\ \text{field-simps},$ 
       $\text{simp}\ \text{add:}\ \text{mult.left-commute}\ \text{semiring-normalization-rules}(18))$ 
  from this 1 show  $\text{gcd}\ (a*b)\ (c*d) = (\text{gcd}\ a\ d) * (\text{gcd}\ b\ c)$  by  $\text{auto}$ 
qed

```

```

lemma  $\text{reindex-product-bij}$ :
  fixes  $a\ b\ m\ k :: \text{nat}$ 
  defines  $S \equiv \{(d1,d2).\ d1\ \text{dvd}\ \text{gcd}\ a\ m \wedge d2\ \text{dvd}\ \text{gcd}\ k\ b\}$ 
  defines  $T \equiv \{d.\ d\ \text{dvd}\ (\text{gcd}\ a\ m) * (\text{gcd}\ k\ b)\}$ 
  defines  $f \equiv (\lambda(d1,d2).\ d1 * d2)$ 
  assumes  $\text{coprime}\ a\ k$ 
  shows  $\text{bij-betw}\ f\ S\ T$ 
  unfolding  $\text{bij-betw-def}$ 
proof

```

```

show inj: inj-on f S
  unfolding f-def
proof -
  {fix d1 d2 d1' d2'
  assume (d1,d2) ∈ S (d1',d2') ∈ S
  then have dvd: d1 dvd gcd a m d2 dvd gcd k b
            d1' dvd gcd a m d2' dvd gcd k b
    unfolding S-def by simp+
  assume f (d1,d2) = f (d1',d2')
  then have eq: d1 * d2 = d1' * d2'
    unfolding f-def by simp
  from eq dvd have eq1: d1 = d1'
    by (simp,meson assms coprime-crossproduct-nat coprime-divisors)
  from eq dvd have eq2: d2 = d2'
    using assms(4) eq1 by auto
  from eq1 eq2 have d1 = d1' ∧ d2 = d2' by simp}
  then show inj-on (λ(d1, d2). d1 * d2) S
    using S-def f-def by (intro inj-onI,blast)
qed
show surj: f ' S = T
proof -
  {fix d
  have d dvd (gcd a m) * (gcd k b)
    ↔ (∃ d1 d2. d = d1*d2 ∧ d1 dvd gcd a m ∧ d2 dvd gcd k b)
    using division-decomp mult-dvd-mono by blast}
  then show ?thesis
    unfolding f-def S-def T-def image-def
    by auto
qed
qed

lemma p-div-set:
shows {p. p ∈ prime-factors a ∧ ¬ p dvd N} =
      ({p. p ∈ prime-factors (a*N)} - {p. p ∈ prime-factors N})
(is ?A = ?B)
proof
show ?A ⊆ ?B
proof (simp)
  { fix p
  assume as: p ∈ # prime-factorization a ∧ ¬ p dvd N
  then have 1: p ∈ prime-factors (a * N)
  proof -
    from in-prime-factors-iff[of p a] as
    have a ≠ 0 p dvd a prime p by simp+
    have N ≠ 0 using (¬ p dvd N) by blast
    have a * N ≠ 0 using ⟨a ≠ 0⟩ ⟨N ≠ 0⟩ by auto
    have p dvd a*N using ⟨p dvd a⟩ by simp
    show ?thesis
      using ⟨a*N ≠ 0⟩ ⟨p dvd a*N⟩ ⟨prime p⟩ in-prime-factors-iff by blast
  }
}

```

```

    qed
    from as have 2:  $p \notin \text{prime-factors } N$  by blast
    from 1 2 have  $p \in \text{prime-factors } (a * N) - \text{prime-factors } N$ 
    by blast
  }
  then show  $\{p. p \in \# \text{ prime-factorization } a \wedge \neg p \text{ dvd } N\}$ 
     $\subseteq \text{prime-factors } (a * N) - \text{prime-factors } N$  by blast
qed

show  $?B \subseteq ?A$ 
proof (simp)
  { fix p
    assume as:  $p \in \text{prime-factors } (a * N) - \text{prime-factors } N$ 
    then have 1:  $\neg p \text{ dvd } N$ 
    proof -
      from as have  $p \in \text{prime-factors } (a * N)$   $p \notin \text{prime-factors } N$ 
      using DiffD1 DiffD2 by blast+
      then show ?thesis by (simp add: in-prime-factors-iff)
    qed
    have 2:  $p \in \# \text{ prime-factorization } a$ 
    proof -
      have  $p \text{ dvd } (a * N)$   $\text{prime } p$   $a * N \neq 0$  using in-prime-factors-iff as by blast+
      have  $p \text{ dvd } a$  using  $\langle \neg p \text{ dvd } N \rangle$  prime-dvd-multD[OF  $\langle \text{prime } p \rangle$   $\langle p \text{ dvd } (a * N) \rangle$ ] by blast
      have  $a \neq 0$  using  $\langle a * N \neq 0 \rangle$  by simp
      show ?thesis using in-prime-factors-iff  $\langle a \neq 0 \rangle$   $\langle p \text{ dvd } a \rangle$   $\langle \text{prime } p \rangle$  by blast
    qed
    from 1 2 have  $p \in \{p. p \in \# \text{ prime-factorization } a \wedge \neg p \text{ dvd } N\}$  by blast
  }
  then show  $\text{prime-factors } (a * N) - \text{prime-factors } N$ 
     $\subseteq \{p. p \in \# \text{ prime-factorization } a \wedge \neg p \text{ dvd } N\}$  by blast
qed
qed

```

lemma *coprime-iff-prime-factors-disjoint:*
fixes $x y :: 'a :: \text{factorial-semiring}$
assumes $x \neq 0$ $y \neq 0$
shows $\text{coprime } x y \iff \text{prime-factors } x \cap \text{prime-factors } y = \{\}$
proof
assume $\text{coprime } x y$
have False **if** $p \in \text{prime-factors } x$ $p \in \text{prime-factors } y$ **for** p
proof -
from *that assms* **have** $p \text{ dvd } x$ $p \text{ dvd } y$
by (auto simp: prime-factors-dvd)
with $\langle \text{coprime } x y \rangle$ **have** $p \text{ dvd } 1$
using coprime-common-divisor **by** auto
with *that assms* **show** False **by** (auto simp: prime-factors-dvd)
qed
thus $\text{prime-factors } x \cap \text{prime-factors } y = \{\}$ **by** auto

```

next
  assume disjoint: prime-factors x  $\cap$  prime-factors y = {}
  show coprime x y
  proof (rule coprimeI)
    fix d assume d: d dvd x d dvd y
    show is-unit d
    proof (rule ccontr)
      assume  $\neg$ is-unit d
      moreover from this and d assms have d  $\neq$  0 by auto
      ultimately obtain p where p: prime p p dvd d
        using prime-divisor-exists by auto
      with d and assms have p  $\in$  prime-factors x  $\cap$  prime-factors y
        by (auto simp: prime-factors-dvd)
      with disjoint show False by auto
    qed
  qed
qed

```

```

lemma coprime-cong-prime-factors:
  fixes x y :: 'a :: factorial-semiring-gcd
  assumes x  $\neq$  0 y  $\neq$  0 x'  $\neq$  0 y'  $\neq$  0
  assumes prime-factors x = prime-factors x'
  assumes prime-factors y = prime-factors y'
  shows coprime x y  $\longleftrightarrow$  coprime x' y'
  using assms by (simp add: coprime-iff-prime-factors-disjoint)

```

```

lemma moebius-prod-not-coprime:
  assumes  $\neg$  coprime N d
  shows moebius-mu (N*d) = 0
proof -
  from assms obtain l where l-form: l dvd N  $\wedge$  l dvd d  $\wedge$   $\neg$  is-unit l
  unfolding coprime-def by blast
  then have l * l dvd N * d using mult-dvd-mono by auto
  then have l2 dvd N*d by (subst power2-eq-square,blast)
  then have  $\neg$  squarefree (N*d)
  unfolding squarefree-def coprime-def using l-form by blast
  then show moebius-mu (N*d) = 0
  using moebius-mu-def by auto
qed

```

Theorem 2.18

```

lemma sum-divisors-moebius-mu-times-multiplicative:
  fixes f :: nat  $\Rightarrow$  'a :: {comm-ring-1}
  assumes multiplicative-function f and n > 0
  shows  $(\sum d \mid d \text{ dvd } n. \text{moebius-mu } d * f d) = (\prod p \in \text{prime-factors } n. 1 - f p)$ 
proof -
  define g where g =  $(\lambda n. \sum d \mid d \text{ dvd } n. \text{moebius-mu } d * f d)$ 
  define g' where g' = dirichlet-prod  $(\lambda n. \text{moebius-mu } n * f n)$   $(\lambda n. \text{if } n = 0 \text{ then } 0 \text{ else } 1)$ 

```

interpret *f*: *multiplicative-function f* **by fact**
have *multiplicative-function* ($\lambda n. \text{if } n = 0 \text{ then } 0 \text{ else } 1 :: 'a$)
by *standard auto*
interpret *multiplicative-function g'* **unfolding** *g'-def*
by (*intro multiplicative-dirichlet-prod multiplicative-function-mult*
moebius-mu.multiplicative-function-axioms assms) *fact+*

have *g'-primepow*: $g' (p \wedge k) = 1 - f p$ **if prime** *p k > 0* **for** *p k*
proof –
have $g' (p \wedge k) = (\sum i \leq k. \text{moebius-mu } (p \wedge i) * f (p \wedge i))$
using *that* **by** (*simp add: g'-def dirichlet-prod-prime-power*)
also have $\dots = (\sum i \in \{0, 1\}. \text{moebius-mu } (p \wedge i) * f (p \wedge i))$
using *that* **by** (*intro sum.mono-neutral-right*) (*auto simp: moebius-mu-power'*)
also have $\dots = 1 - f p$
using *that* **by** (*simp add: moebius-mu.prime*)
finally show *?thesis* .
qed

have $g' n = g n$
by (*simp add: g-def g'-def dirichlet-prod-def*)
also from *assms* **have** $g' n = (\prod p \in \text{prime-factors } n. g' (p \wedge \text{multiplicity } p n))$
by (*intro prod-prime-factors*) *auto*
also have $\dots = (\prod p \in \text{prime-factors } n. 1 - f p)$
by (*intro prod.cong*) (*auto simp: g'-primepow prime-factors-multiplicity*)
finally show *?thesis* **by** (*simp add: g-def*)
qed

lemma *multiplicative-ind-coprime* [*intro*]: *multiplicative-function* (*ind (coprime N)*)
by (*intro multiplicative-function-ind*) *auto*

lemma *sum-divisors-moebius-mu-times-multiplicative-revisited*:
fixes $f :: \text{nat} \Rightarrow 'a :: \{\text{comm-ring-1}\}$
assumes *multiplicative-function* $f n > 0$ $N > 0$
shows $(\sum d \mid d \text{ dvd } n \wedge \text{coprime } N d. \text{moebius-mu } d * f d) =$
 $(\prod p \in \{p. p \in \text{prime-factors } n \wedge \neg (p \text{ dvd } N)\}. 1 - f p)$
proof –
have $(\sum d \mid d \text{ dvd } n \wedge \text{coprime } N d. \text{moebius-mu } d * f d) =$
 $(\sum d \mid d \text{ dvd } n. \text{moebius-mu } d * (\text{ind } (\text{coprime } N) d * f d))$
using *assms* **by** (*intro sum.mono-neutral-cong-left*) (*auto simp: ind-def*)
also have $\dots = (\prod p \in \text{prime-factors } n. 1 - \text{ind } (\text{coprime } N) p * f p)$
using *assms* **by** (*intro sum-divisors-moebius-mu-times-multiplicative*)
(auto intro: multiplicative-function-mult)
also from *assms* **have** $\dots = (\prod p \mid p \in \text{prime-factors } n \wedge \neg (p \text{ dvd } N). 1 - f p)$
by (*intro prod.mono-neutral-cong-right*)
(auto simp: ind-def prime-factors-dvd coprime-commute dest: prime-imp-coprime)
finally show *?thesis* .
qed

1.2 Neutral element of the Dirichlet product

definition *dirichlet-prod-neutral* $n = (if\ n = 1\ then\ 1\ else\ 0)$ for $n :: nat$

lemma *dirichlet-prod-neutral-intro*:

fixes $S :: nat \Rightarrow complex$ **and** $f :: nat \Rightarrow nat \Rightarrow complex$

defines $S \equiv (\lambda(n::nat). (\sum k \mid k \in \{1..n\} \wedge coprime\ k\ n. (f\ k\ n)))$

shows $S(n) = (\sum k \in \{1..n\}. f\ k\ n * dirichlet-prod-neutral\ (gcd\ k\ n))$

proof –

let $?g = \lambda k. (f\ k\ n) * (dirichlet-prod-neutral\ (gcd\ k\ n))$

have zeros: $\forall k \in \{1..n\} - \{k. k \in \{1..n\} \wedge coprime\ k\ n\}. ?g\ k = 0$

proof

fix k

assume $k \in \{1..n\} - \{k \in \{1..n\}. coprime\ k\ n\}$

then show $(f\ k\ n) * dirichlet-prod-neutral\ (gcd\ k\ n) = 0$

by (*simp add: dirichlet-prod-neutral-def*[of $gcd\ k\ n$] *split: if-splits,presburger*)

qed

have $S\ n = (\sum k \mid k \in \{1..n\} \wedge coprime\ k\ n. (f\ k\ n))$

by (*simp add: S-def*)

also have $\dots = sum\ ?g\ \{k. k \in \{1..n\} \wedge coprime\ k\ n\}$

by (*simp add: dirichlet-prod-neutral-def split: if-splits*)

also have $\dots = sum\ ?g\ \{1..n\}$

by (*intro sum.mono-neutral-left, auto simp add: zeros*)

finally show *?thesis* **by** *blast*

qed

lemma *dirichlet-prod-neutral-right-neutral*:

dirichlet-prod $f\ dirichlet-prod-neutral\ n = f\ n$ **if** $n > 0$ **for** $f :: nat \Rightarrow complex$

and n

proof –

{fix $d :: nat$

assume $d\ dvd\ n$

then have *eq*: $n = d \longleftrightarrow n\ div\ d = 1$

using *div-self that dvd-mult-div-cancel* **by** *force*

have $f(d) * dirichlet-prod-neutral(n\ div\ d) = (if\ n = d\ then\ f(d)\ else\ 0)$

by (*simp add: dirichlet-prod-neutral-def eq*)}

note *summand = this*

have *dirichlet-prod* $f\ dirichlet-prod-neutral\ n =$

$(\sum d \mid d\ dvd\ n. f(d) * dirichlet-prod-neutral(n\ div\ d))$

unfolding *dirichlet-prod-def* **by** *blast*

also have $\dots = (\sum d \mid d\ dvd\ n. (if\ n = d\ then\ f(d)\ else\ 0))$

using *summand* **by** *simp*

also have $\dots = (\sum d \mid d = n. (if\ n = d\ then\ f(d)\ else\ 0))$

using *that* **by** (*intro sum.mono-neutral-right, auto*)

also have $\dots = f(n)$ **by** *simp*

finally show *?thesis* **by** *simp*

qed

lemma *dirichlet-prod-neutral-left-neutral*:
dirichlet-prod dirichlet-prod-neutral f n = f n
if $n > 0$ **for** $f :: \text{nat} \Rightarrow \text{complex}$ **and** n
using *dirichlet-prod-neutral-right-neutral*[*OF that, of f*]
dirichlet-prod-commutes[*of f dirichlet-prod-neutral*]
by *argo*

corollary *I-right-neutral-0*:
fixes $f :: \text{nat} \Rightarrow \text{complex}$
assumes $f\ 0 = 0$
shows *dirichlet-prod f dirichlet-prod-neutral n = f n*
using *assms dirichlet-prod-neutral-right-neutral* **by** (*cases n, simp, blast*)

1.3 Multiplicative functions

lemma *mult-id: multiplicative-function id*
by (*simp add: multiplicative-function-def*)

lemma *mult-moebius: multiplicative-function moebius-mu*
using *Moebius-Mu.moebius-mu.multiplicative-function-axioms*
by *simp*

lemma *mult-of-nat: multiplicative-function of-nat*
using *multiplicative-function-def of-nat-0 of-nat-1 of-nat-mult* **by** *blast*

lemma *mult-of-nat-c: completely-multiplicative-function of-nat*
by (*simp add: completely-multiplicative-function-def*)

lemma *completely-multiplicative-nonzero*:
fixes $f :: \text{nat} \Rightarrow \text{complex}$
assumes *completely-multiplicative-function f*
 $d \neq 0$
 $\bigwedge p. \text{prime } p \implies f(p) \neq 0$
shows $f(d) \neq 0$
using *assms(2)*

proof (*induction d rule: nat-less-induct*)
case ($1\ n$)
then show *?case*
proof (*cases n = 1*)
case *True*
then show *?thesis*
using *assms(1)*
unfolding *completely-multiplicative-function-def* **by** *simp*

next
case *False*
then obtain p **where** $2:\text{prime } p \wedge p\ \text{dvd } n$
using *prime-factor-nat* **by** *blast*
then obtain a **where** $3: n = p * a\ a \neq 0$
using 1 **by** *auto*

then have 4: $f(a) \neq 0$ **using** 1
using 2 *prime-nat-iff* **by** *fastforce*
have 5: $f(p) \neq 0$ **using** *assms(3)* 2 **by** *simp*
from 3 4 5 **show** *?thesis*
by (*simp add: assms(1) completely-multiplicative-function.mult*)
qed
qed

lemma *multipl-div*:
fixes $m\ k\ d1\ d2 :: \text{nat}$ **and** $f :: \text{nat} \Rightarrow \text{complex}$
assumes *multiplicative-function* $f\ d1\ dvd\ m\ d2\ dvd\ k\ coprime\ m\ k$
shows $f\ ((m*k)\ div\ (d1*d2)) = f(m\ div\ d1) * f(k\ div\ d2)$
using *assms*
unfolding *multiplicative-function-def*
using *assms(1) multiplicative-function.mult-coprime* **by** *fastforce*

lemma *multipl-div-mono*:
fixes $m\ k\ d :: \text{nat}$ **and** $f :: \text{nat} \Rightarrow \text{complex}$
assumes *completely-multiplicative-function* f
 $d\ dvd\ k\ d > 0$
 $\bigwedge p.\ prime\ p \implies f(p) \neq 0$
shows $f\ (k\ div\ d) = f(k)\ div\ f(d)$
proof –
have $d \neq 0$ **using** *assms(2,3)* **by** *auto*
then have $nz: f(d) \neq 0$ **using** *assms(1,4) completely-multiplicative-nonzero* **by**
simp

from *assms(2,3)* **obtain** a **where** $div: k = a * d$ **by** *fastforce*
have $f\ (k\ div\ d) = f\ ((a*d)\ div\ d)$ **using** *div* **by** *simp*
also have $\dots = f(a)$ **using** *assms(3) div* **by** *simp*
also have $\dots = f(a)*f(d)\ div\ f(d)$ **using** *nz* **by** *auto*
also have $\dots = f(a*d)\ div\ f(d)$
by (*simp add: div assms(1) completely-multiplicative-function.mult*)
also have $\dots = f\ (k)\ div\ f(d)$ **using** *div* **by** *simp*
finally show *?thesis* **by** *simp*
qed

lemma *comp-to-mult*: *completely-multiplicative-function* $f \implies$
multiplicative-function f
unfolding *completely-multiplicative-function-def*
multiplicative-function-def **by** *auto*

end

2 Periodic arithmetic functions

theory *Periodic-Arithmetic*
imports
Complex-Main

HOL-Number-Theory.Cong
begin

definition

periodic-arithmetic $f\ k = (\forall n. f\ (n+k) = f\ n)$
for $n :: \text{int}$ **and** $k :: \text{nat}$ **and** $f :: \text{nat} \Rightarrow \text{complex}$

lemma *const-periodic-arithmetic*: *periodic-arithmetic* $(\lambda x. y)$ k
unfolding *periodic-arithmetic-def* **by** *blast*

lemma *add-periodic-arithmetic*:

fixes $f\ g :: \text{nat} \Rightarrow \text{complex}$
assumes *periodic-arithmetic* $f\ k$
assumes *periodic-arithmetic* $g\ k$
shows *periodic-arithmetic* $(\lambda n. f\ n + g\ n)$ k
using *assms* **unfolding** *periodic-arithmetic-def* **by** *simp*

lemma *mult-periodic-arithmetic*:

fixes $f\ g :: \text{nat} \Rightarrow \text{complex}$
assumes *periodic-arithmetic* $f\ k$
assumes *periodic-arithmetic* $g\ k$
shows *periodic-arithmetic* $(\lambda n. f\ n * g\ n)$ k
using *assms* **unfolding** *periodic-arithmetic-def* **by** *simp*

lemma *scalar-mult-periodic-arithmetic*:

fixes $f :: \text{nat} \Rightarrow \text{complex}$ **and** $a :: \text{complex}$
assumes *periodic-arithmetic* $f\ k$
shows *periodic-arithmetic* $(\lambda n. a * f\ n)$ k
using *mult-periodic-arithmetic[OF const-periodic-arithmetic[of a k] assms(1)]* **by**
simp

lemma *fin-sum-periodic-arithmetic-set*:

fixes $f\ g :: \text{nat} \Rightarrow \text{complex}$
assumes $\forall i \in A. \text{periodic-arithmetic}\ (h\ i)\ k$
shows *periodic-arithmetic* $(\lambda n. \sum i \in A. h\ i\ n)$ k
using *assms* **by** (*simp add: periodic-arithmetic-def*)

lemma *mult-period*:

assumes *periodic-arithmetic* $g\ k$
shows *periodic-arithmetic* $g\ (k*q)$
using *assms*

proof (*induction* q)

case 0 **then show** ?*case* **unfolding** *periodic-arithmetic-def* **by** *simp*

next

case (*Suc* m)

then show ?*case*

unfolding *periodic-arithmetic-def*

proof –

{ **fix** n

```

    have  $g (n + k * Suc m) = g (n + k + k * m)$ 
      by (simp add: algebra-simps)
    also have  $\dots = g(n)$ 
      using Suc.IH[OF Suc.prem] assms
      unfolding periodic-arithmetic-def by simp
    finally have  $g (n + k * Suc m) = g(n)$  by blast
  }
  then show  $\forall n. g (n + k * Suc m) = g n$  by auto
qed
qed

```

lemma *unique-periodic-arithmetic-extension*:

```

assumes  $k > 0$ 
assumes  $\forall j < k. g j = h j$ 
assumes periodic-arithmetic  $g$   $k$  and periodic-arithmetic  $h$   $k$ 
shows  $g i = h i$ 
proof (cases  $i < k$ )
  case True then show ?thesis using assms by simp
next
  case False then show ?thesis
  proof -
    have  $k * (i \text{ div } k) + (i \text{ mod } k) = i \wedge (i \text{ mod } k) < k$ 
      by (simp add: assms(1) algebra-simps)
    then obtain  $q r$  where euclid-div:  $k*q + r = i \wedge r < k$ 
      using mult.commute by blast
    from assms(3) assms(4)
    have periodic-arithmetic  $g (k*q)$  periodic-arithmetic  $h (k*q)$ 
      using mult-period by simp+
    have  $g(k*q+r) = g(r)$ 
      using  $\langle$ periodic-arithmetic  $g (k*q)$  $\rangle$  unfolding periodic-arithmetic-def
      using add commute[of  $k*q r$ ] by presburger
    also have  $\dots = h(r)$ 
      using euclid-div assms(2) by simp
    also have  $\dots = h(k*q+r)$ 
      using  $\langle$ periodic-arithmetic  $h (k*q)$  $\rangle$  add commute[of  $k*q r$ ]
      unfolding periodic-arithmetic-def by presburger
    also have  $\dots = h(i)$  using euclid-div by simp
    finally show  $g(i) = h(i)$  using euclid-div by simp
  qed
qed
qed

```

lemma *periodic-arithmetic-sum-periodic-arithmetic*:

```

assumes periodic-arithmetic  $f$   $k$ 
shows  $(\sum l \in \{m..n\}. f l) = (\sum l \in \{m+k..n+k\}. f l)$ 
using periodic-arithmetic-def assms
by (intro sum.reindex-bij-witness
    [of  $\{m..n\} \lambda l. l-k \lambda l. l+k \{m+k..n+k\} f f])
    auto$ 
```

lemma *mod-periodic-arithmetic*:
fixes $n\ m :: \text{nat}$
assumes *periodic-arithmetic* $f\ k$
assumes $n \bmod k = m \bmod k$
shows $f\ n = f\ m$
proof –
obtain q **where** $1: n = q*k + (n \bmod k)$
using *div-mult-mod-eq*[*of n k, symmetric*] **by** *blast*
obtain q' **where** $2: m = q'*k + (m \bmod k)$
using *div-mult-mod-eq*[*of m k, symmetric*] **by** *blast*
from 1 **have** $f\ n = f\ (q*k + (n \bmod k))$ **by** *auto*
also have $\dots = f\ (n \bmod k)$
using *mult-period*[*of f k q*] *assms(1)* *periodic-arithmetic-def*[*of f k*q*]
by (*simp add: algebra-simps, subst add.commute, blast*)
also have $\dots = f\ (m \bmod k)$ **using** *assms(2)* **by** *auto*
also have $\dots = f\ (q'*k + (m \bmod k))$
using *mult-period*[*of f k q'*] *assms(1)* *periodic-arithmetic-def*[*of f k*q'*]
by (*simp add: algebra-simps, subst add.commute, presburger*)
also have $\dots = f\ m$ **using** 2 **by** *auto*
finally show $f\ n = f\ m$ **by** *simp*
qed

lemma *cong-periodic-arithmetic*:
assumes *periodic-arithmetic* $f\ k\ [a = b] \pmod k$
shows $f\ a = f\ b$
using *assms mod-periodic-arithmetic*[*of f k a b*] **by** (*auto simp: cong-def*)

lemma *cong-nat-imp-eq*:
fixes $m :: \text{nat}$
assumes $m > 0\ x \in \{a..<a+m\}\ y \in \{a..<a+m\}\ [x = y] \pmod m$
shows $x = y$
using *assms*
proof (*induction x y rule: linorder-wlog*)
case (*le x y*)
have $[y - x = 0] \pmod m$
using *cong-diff-iff-cong-0-nat cong-sym le* **by** *blast*
thus $x = y$
using *le* **by** (*auto simp: cong-def*)
qed (*auto simp: cong-sym*)

lemma *inj-on-mod-nat*:
fixes $m :: \text{nat}$
assumes $m > 0$
shows *inj-on* $(\lambda x. x \bmod m)\ \{a..<a+m\}$
proof
fix $x\ y$ **assume** $xy: x \in \{a..<a+m\}\ y \in \{a..<a+m\}$ **and** $eq: x \bmod m = y \bmod m$
from $\langle m > 0 \rangle$ **and** xy **show** $x = y$
by (*rule cong-nat-imp-eq*) (*use eq in (simp-all add: cong-def)*)

qed

lemma *bij-betw-mod-nat-atLeastLessThan*:

fixes $k\ d :: \text{nat}$

assumes $k > 0$

defines $g \equiv (\lambda i. \text{nat } ((\text{int } i - \text{int } d) \bmod \text{int } k) + d)$

shows $\text{bij-betw } (\lambda i. i \bmod k) \{d..<d+k\} \{..<k\}$

unfolding *bij-betw-def*

proof

show $\text{inj: inj-on } (\lambda i. i \bmod k) \{d..<d+k\}$

by (*rule inj-on-mod-nat*) *fact+*

have $(\lambda i. i \bmod k) ' \{d..<d+k\} \subseteq \{..<k\}$

by *auto*

moreover have $\text{card } ((\lambda i. i \bmod k) ' \{d..<d+k\}) = \text{card } \{..<k\}$

using *inj* **by** (*subst card-image*) *auto*

ultimately show $(\lambda i. i \bmod k) ' \{d..<d+k\} = \{..<k\}$

by (*intro card-subset-eq*) *auto*

qed

lemma *periodic-arithmetic-sum-periodic-arithmetic-shift*:

fixes $k\ d :: \text{nat}$

assumes *periodic-arithmetic* $f\ k\ k > 0\ d > 0$

shows $(\sum l \in \{0..k-1\}. f\ l) = (\sum l \in \{d..d+k-1\}. f\ l)$

proof –

have $(\sum l \in \{0..k-1\}. f\ l) = (\sum l \in \{0..<k\}. f\ l)$

using *assms(2)* **by** (*intro sum.cong*) *auto*

also have $\dots = (\sum l \in \{d..<d+k\}. f\ (l \bmod k))$

using *assms(2)*

by (*simp add: sum.reindex-bij-betw*[*OF bij-betw-mod-nat-atLeastLessThan*[*of k d*]]

lessThan-atLeast0)

also have $\dots = (\sum l \in \{d..<d+k\}. f\ l)$

using *mod-periodic-arithmetic*[*of f k*] *assms(1)* *sum.cong*

by (*meson mod-mod-trivial*)

also have $\dots = (\sum l \in \{d..d+k-1\}. f\ l)$

using *assms(2,3)* **by** (*intro sum.cong*) *auto*

finally show *?thesis* **by** *auto*

qed

lemma *self-bij-0-k*:

fixes $a\ k :: \text{nat}$

assumes *coprime* $a\ k$ [$a*i = 1 \pmod k$] $k > 0$

shows $\text{bij-betw } (\lambda r. r*a \bmod k) \{0..k-1\} \{0..k-1\}$

unfolding *bij-betw-def*

proof

show $\text{inj-on } (\lambda r. r*a \bmod k) \{0..k-1\}$

proof –

{fix $r1\ r2$

assume *in-k*: $r1 \in \{0..k-1\}\ r2 \in \{0..k-1\}$

```

assume as: [ $r1*a = r2*a$ ] (mod k)
then have [ $r1*a*i = r2*a*i$ ] (mod k)
  using cong-scalar-right by blast
then have [ $r1 = r2$ ] (mod k)
  using cong-mult-rcancel-nat as assms(1) by simp
then have  $r1 = r2$  using in-k
  using assms(3) cong-less-modulus-unique-nat by auto}
note eq = this
show ?thesis unfolding inj-on-def
  by (safe, simp add: eq cong-def)
qed
define f where  $f = (\lambda r. r * a \text{ mod } k)$ 
show  $f \text{ ` } \{0..k - 1\} = \{0..k - 1\}$ 
  unfolding image-def
proof (standard)
  show  $\{y. \exists x \in \{0..k - 1\}. y = f x\} \subseteq \{0..k - 1\}$ 
  proof -
    {fix y
      assume  $y \in \{y. \exists x \in \{0..k - 1\}. y = f x\}$ 
      then obtain x where  $y = f x$  by blast
      then have  $y \in \{0..k-1\}$ 
        unfolding f-def
        using Suc-pred assms(3) lessThan-Suc-atMost by fastforce}
    then show ?thesis by blast
  qed
show  $\{0..k - 1\} \subseteq \{y. \exists x \in \{0..k - 1\}. y = f x\}$ 
proof -
  { fix x
    assume ass:  $x \in \{0..k-1\}$ 
    then have  $x * i \text{ mod } k \in \{0..k-1\}$ 
    proof -
      have  $x * i \text{ mod } k \in \{0..<k\}$  by (simp add: assms(3))
      have  $\{0..<k\} = \{0..k-1\}$  using Suc-diff-1 assms(3) by auto
      show ?thesis using  $\langle x * i \text{ mod } k \in \{0..<k\} \rangle \langle \{0..<k\} = \{0..k-1\} \rangle$  by blast
    qed
    then have  $f (x * i \text{ mod } k) = x$ 
    proof -
      have  $f (x * i \text{ mod } k) = (x * i \text{ mod } k) * a \text{ mod } k$ 
        unfolding f-def by blast
      also have  $\dots = (x*i*a) \text{ mod } k$ 
        by (simp add: mod-mult-left-eq)
      also have  $\dots = (x*1) \text{ mod } k$ 
        using assms(2)
        unfolding cong-def
        by (subst mult.assoc, subst (2) mult.commute,
          subst mod-mult-right-eq[symmetric],simp)
      also have  $\dots = x$  using ass assms(3) by auto
      finally show ?thesis .
    qed
  }

```



```

    then have  $x \in \{y. \exists x \in \{0..k-1\}. y = f x\}$ 
      using  $\langle x * i \bmod k \in \{0..k-1\} \rangle$  by force
  }
  then show ?thesis by blast
qed
qed
qed

```

lemma *periodic-arithmetic-homothecy*:

```

  assumes periodic-arithmetic  $f$   $k$ 
  shows periodic-arithmetic  $(\lambda l. f (l*a))$   $k$ 
  unfolding periodic-arithmetic-def
proof
  fix  $n$ 
  have  $f ((n + k) * a) = f(n*a+k*a)$  by (simp add: algebra-simps)
  also have  $\dots = f(n*a)$ 
    using mult-period[OF assms] unfolding periodic-arithmetic-def by simp
  finally show  $f ((n + k) * a) = f (n * a)$  by simp
qed

```

theorem *periodic-arithmetic-remove-homothecy*:

```

  assumes coprime  $a$   $k$  periodic-arithmetic  $f$   $k$   $k > 0$ 
  shows  $(\sum l=1..k. f l) = (\sum l=1..k. f (l*a))$ 
proof -
  obtain  $i$  where inv:  $[a*i = 1] \pmod k$ 
    using assms(1) coprime-iff-invertible-nat[of  $a$   $k$ ] by auto
  from this self-bij-0-k assms
  have bij: bij-betw  $(\lambda r. r * a \bmod k)$   $\{0..k-1\}$   $\{0..k-1\}$  by blast

  have  $(\sum l = 1..k. f(l)) = (\sum l = 0..k-1. f(l))$ 
    using periodic-arithmetic-sum-periodic-arithmetic-shift[of  $f$   $k$  1] assms by simp
  also have  $\dots = (\sum l = 0..k-1. f(l*a \bmod k))$ 
    using sum.reindex-bij-betw[OF bij,symmetric] by blast
  also have  $\dots = (\sum l = 0..k-1. f(l*a))$ 
    by (intro sum.cong refl) (use mod-periodic-arithmetic[OF assms(2)] mod-mod-trivial
  in blast)
  also have  $\dots = (\sum l = 1..k. f(l*a))$ 
    using periodic-arithmetic-sum-periodic-arithmetic-shift[of  $(\lambda l. f(l*a))$   $k$  1]
      periodic-arithmetic-homothecy[OF assms(2)] assms(3) by fastforce
  finally show ?thesis by blast
qed

```

end

theory *Complex-Roots-Of-Unity*

imports

HOL-Analysis.Analysis

Periodic-Arithmetic

begin

3 Complex roots of unity

definition

$unity_root\ k\ n = cis\ (2 * pi * of_int\ n / of_nat\ k)$

lemma

$unity_root\ k\ 0 [simp]: unity_root\ k\ 0 = 1$ **and**

$unity_root\ 0\ n [simp]: unity_root\ 0\ n = 1$

unfolding $unity_root_def$ **by** $simp+$

lemma $unity_root_conv_exp$:

$unity_root\ k\ n = exp\ (of_real\ (2*pi*n/k) * i)$

unfolding $unity_root_def$

by $(subst\ cis_conv_exp, subst\ mult.commute, blast)$

lemma $unity_root_mod$:

$unity_root\ k\ (n\ mod\ int\ k) = unity_root\ k\ n$

proof $(cases\ k = 0)$

case $True$ **then show** $?thesis$ **by** $simp$

next

case $False$

obtain $q :: int$ **where** $q_def: n = q*k + (n\ mod\ k)$

using $div_mult_mod_eq[symmetric]$ **by** $blast$

have $n / k = q + (n\ mod\ k) / k$

proof $(auto\ simp\ add: divide_simps\ False)$

have $real_of_int\ n = real_of_int\ (q*k + (n\ mod\ k))$

using q_def **by** $simp$

also have $\dots = real_of_int\ q * real\ k + real_of_int\ (n\ mod\ k)$

using $of_int_add\ of_int_mult$ **by** $simp$

finally show $real_of_int\ n = real_of_int\ q * real\ k + real_of_int\ (n\ mod\ k)$

by $blast$

qed

then have $(2*pi*n/k) = 2*pi*q + (2*pi*(n\ mod\ k)/k)$

using $False$ **by** $(auto\ simp\ add: field_simps)$

then have $(2*pi*n/k)*i = 2*pi*q*i + (2*pi*(n\ mod\ k)/k)*i$ **(is** $?l = ?r1 + ?r2$)

by $(auto\ simp\ add: algebra_simps)$

then have $exp\ ?l = exp\ ?r2$

using exp_plus_2pin **by** $(simp\ add: exp_add\ mult.commute)$

then show $?thesis$

using $unity_root_def\ unity_root_conv_exp$ **by** $simp$

qed

lemma $unity_root_cong$:

assumes $[m = n]\ (mod\ int\ k)$

shows $unity_root\ k\ m = unity_root\ k\ n$

proof $-$

from $assms$ **have** $m\ mod\ int\ k = n\ mod\ int\ k$

by $(auto\ simp: cong_def)$

hence $unity_root\ k\ (m\ mod\ int\ k) = unity_root\ k\ (n\ mod\ int\ k)$

by *simp*
 thus *?thesis* by (*simp add: unity-root-mod*)
 qed

lemma *unity-root-mod-nat*:
 $unity_root\ k\ (nat\ (n\ mod\ int\ k)) = unity_root\ k\ n$
proof (*cases k*)
 case (*Suc l*)
 then have $n\ mod\ int\ k \geq 0$ by *auto*
 show *?thesis*
 unfolding *int-nat-eq*
 by (*simp add: <n mod int k ≥ 0> unity-root-mod*)
 qed *auto*

lemma *unity-root-eqD*:
 assumes *gr*: $k > 0$
 assumes *eq*: $unity_root\ k\ i = unity_root\ k\ j$
 shows $i\ mod\ k = j\ mod\ k$
proof –
 let *?arg1* = $(2*pi*i/k)*i$
 let *?arg2* = $(2*pi*j/k)*i$
 from *eq* *unity-root-conv-exp* have $exp\ ?arg1 = exp\ ?arg2$ by *simp*
 from *this* *exp-eq*
 obtain *n* :: *int* where $?arg1 = ?arg2 + (2*n*pi)*i$ by *blast*
 then have *e1*: $?arg1 - ?arg2 = 2*n*pi*i$ by *simp*
 have *e2*: $?arg1 - ?arg2 = 2*(i-j)*(1/k)*pi*i$
 by (*auto simp add: algebra-simps*)
 from *e1 e2* have $2*n*pi*i = 2*(i-j)*(1/k)*pi*i$ by *simp*
 then have $2*n*k*pi*i = 2*(i-j)*pi*i$
 by (*simp add: divide-simps <k > 0>(simp add: field-simps)*)
 then have $2*n*k = 2*(i-j)$
 by (*meson complex-i-not-zero mult-cancel-right of-int-eq-iff of-real-eq-iff pi-neq-zero*)
 then have $n*k = i-j$ by *auto*
 then show *?thesis* by *Groebner-Basis.algebra*
 qed

lemma *unity-root-eq-1-iff*:
 fixes *k n* :: *nat*
 assumes $k > 0$
 shows $unity_root\ k\ n = 1 \longleftrightarrow k\ dvd\ n$
proof –
 have $unity_root\ k\ n = exp\ ((2*pi*n/k)*i)$
 by (*simp add: unity-root-conv-exp*)
 also have $exp\ ((2*pi*n/k)*i) = 1 \longleftrightarrow k\ dvd\ n$
 using *complex-root-unity-eq-1[of k n]* *assms*
 by (*auto simp add: algebra-simps*)
 finally show *?thesis* by *simp*
 qed

lemma *unity-root-pow*: $\text{unity-root } k \ n \ \hat{=} \ m = \text{unity-root } k \ (n * m)$
using *unity-root-def*
by (*simp add: Complex.DeMoivre mult.commute algebra-split-simps(6)*)

lemma *unity-root-add*: $\text{unity-root } k \ (m + n) = \text{unity-root } k \ m * \text{unity-root } k \ n$
by (*simp add: unity-root-conv-exp add-divide-distrib algebra-simps exp-add*)

lemma *unity-root-uminus*: $\text{unity-root } k \ (-m) = \text{cnj} \ (\text{unity-root } k \ m)$
unfolding *unity-root-conv-exp exp-cnj* **by** *simp*

lemma *inverse-unity-root*: $\text{inverse} \ (\text{unity-root } k \ m) = \text{cnj} \ (\text{unity-root } k \ m)$
unfolding *unity-root-conv-exp exp-cnj* **by** (*simp add: field-simps exp-minus*)

lemma *unity-root-diff*: $\text{unity-root } k \ (m - n) = \text{unity-root } k \ m * \text{cnj} \ (\text{unity-root } k \ n)$
using *unity-root-add[of k m -n]* **by** (*simp add: unity-root-uminus*)

lemma *unity-root-eq-1-iff-int*:
fixes $k :: \text{nat}$ **and** $n :: \text{int}$
assumes $k > 0$
shows $\text{unity-root } k \ n = 1 \iff k \ \text{dvd} \ n$
proof (*cases n ≥ 0*)
case *True*
obtain n' **where** $n = \text{int } n'$
using *zero-le-imp-eq-int[OF True]* **by** *blast*
then show *?thesis*
using *unity-root-eq-1-iff[OF ⟨k > 0⟩, of n'] of-nat-dvd-iff* **by** *blast*
next
case *False*
then have $-n \geq 0$ **by** *auto*
have $\text{unity-root } k \ n = \text{inverse} \ (\text{unity-root } k \ (-n))$
unfolding *inverse-unity-root* **by** (*simp add: unity-root-uminus*)
then have $(\text{unity-root } k \ n = 1) = (\text{unity-root } k \ (-n) = 1)$
by *simp*
also have $(\text{unity-root } k \ (-n) = 1) = (k \ \text{dvd} \ (-n))$
using *unity-root-eq-1-iff[of k nat (-n), OF ⟨k > 0⟩ False int-dvd-int-iff[of k nat (-n)] nat-0-le[OF ⟨-n ≥ 0⟩]* **by** *auto*
finally show *?thesis* **by** *simp*
qed

lemma *unity-root-eq-1 [simp]*: $\text{int } k \ \text{dvd} \ n \implies \text{unity-root } k \ n = 1$
by (*cases k = 0*) (*auto simp: unity-root-eq-1-iff-int*)

lemma *unity-periodic-arithmetic*:
periodic-arithmetic (unity-root k) k
unfolding *periodic-arithmetic-def*
proof
fix n
have $\text{unity-root } k \ (n + k) = \text{unity-root } k \ ((n+k) \bmod k)$

using *unity-root-mod*[of k] *zmod-int* **by** *presburger*
also have $\text{unity-root } k ((n+k) \text{ mod } k) = \text{unity-root } k n$
using *unity-root-mod* *zmod-int* **by** *auto*
finally show $\text{unity-root } k (n + k) = \text{unity-root } k n$ **by** *simp*
qed

lemma *unity-periodic-arithmetic-mult*:

periodic-arithmetic ($\lambda n. \text{unity-root } k (m * \text{int } n)$) k

unfolding *periodic-arithmetic-def*

proof

fix n

have $\text{unity-root } k (m * \text{int } (n + k)) =$

$\text{unity-root } k (m*n + m*k)$

by (*simp add: algebra-simps*)

also have $\dots = \text{unity-root } k (m*n)$

using *unity-root-mod*[of k $m * \text{int } n$] *unity-root-mod*[of k $m * \text{int } n + m * \text{int } k$]

mod-mult-self3 **by** *presburger*

finally show $\text{unity-root } k (m * \text{int } (n + k)) =$

$\text{unity-root } k (m * \text{int } n)$ **by** *simp*

qed

lemma *unity-root-periodic-arithmetic-mult-minus*:

shows *periodic-arithmetic* ($\lambda i. \text{unity-root } k (-\text{int } i * \text{int } m)$) k

unfolding *periodic-arithmetic-def*

proof

fix n

have $\text{unity-root } k (-(n + k) * m) = \text{cnj } (\text{unity-root } k (n*m + k*m))$

by (*simp add: ring-distrib unity-root-diff unity-root-add unity-root-uminus*)

also have $\dots = \text{cnj } (\text{unity-root } k (n*m))$

using *mult-period*[of *unity-root* k k m] *unity-periodic-arithmetic*[of k]

unfolding *periodic-arithmetic-def* **by** *presburger*

also have $\dots = \text{unity-root } k (-n*m)$

by (*simp add: unity-root-uminus*)

finally show $\text{unity-root } k (-(n + k) * m) = \text{unity-root } k (-n*m)$

by *simp*

qed

lemma *unity-div*:

fixes $a :: \text{int}$ **and** $d :: \text{nat}$

assumes $d \text{ dvd } k$

shows $\text{unity-root } k (a*d) = \text{unity-root } (k \text{ div } d) a$

proof –

have $1: (2*pi*(a*d)/k) = (2*pi*a)/(k \text{ div } d)$

using *Suc-pred assms* **by** (*simp add: divide-simps, fastforce*)

have $\text{unity-root } k (a*d) = \text{exp } ((2*pi*(a*d)/k)* i)$

using *unity-root-conv-exp* **by** *simp*

also have $\dots = \text{exp } (((2*pi*a)/(k \text{ div } d))* i)$

using 1 **by** *simp*

also have ... = *unity-root* (k div d) a
 using *unity-root-conv-exp* by *simp*
 finally show ?thesis by *simp*
 qed

lemma *unity-div-num*:
 assumes $k > 0$ $d > 0$ $d \text{ dvd } k$
 shows *unity-root* k (x * (k div d)) = *unity-root* d x
 using *assms dvd-div-mult-self unity-div* by *auto*

4 Geometric sums of roots of unity

Apostol calls these ‘geometric sums’, which is a bit too generic. We therefore decided to refer to them as ‘sums of roots of unity’.

definition *unity-root-sum* k n = $(\sum m < k. \text{unity-root } k (n * \text{of-nat } m))$

lemma *unity-root-sum-0-left* [*simp*]: *unity-root-sum* 0 n = 0 **and**
unity-root-sum-0-right [*simp*]: $k > 0 \implies \text{unity-root-sum } k 0 = k$
unfolding *unity-root-sum-def* by *simp-all*

Theorem 8.1

theorem *unity-root-sum*:
 fixes $k :: \text{nat}$ **and** $n :: \text{int}$
 assumes *gr*: $k \geq 1$
 shows $k \text{ dvd } n \implies \text{unity-root-sum } k n = k$
 and $\neg k \text{ dvd } n \implies \text{unity-root-sum } k n = 0$

proof –

assume *dvd*: $k \text{ dvd } n$
 let $?x = \text{unity-root } k n$
 have *unit*: $?x = 1$ using *dvd gr unity-root-eq-1-iff-int* by *auto*
 have *exp*: $?x^m = \text{unity-root } k (n * m)$ for m using *unity-root-pow* by *simp*
 have *unity-root-sum* $k n = (\sum m < k. \text{unity-root } k (n * m))$
 using *unity-root-sum-def* by *simp*
 also have ... = $(\sum m < k. ?x^m)$ using *exp* by *auto*
 also have ... = $(\sum m < k. 1)$ using *unit* by *simp*
 also have ... = k using *gr* by (*induction* k, *auto*)
 finally show *unity-root-sum* $k n = k$ by *simp*

next

assume *dvd*: $\neg k \text{ dvd } n$
 let $?x = \text{unity-root } k n$
 have $?x \neq 1$ using *dvd gr unity-root-eq-1-iff-int* by *auto*
 have $(?x^k - 1) / (?x - 1) = (\sum m < k. ?x^m)$
 using *geometric-sum*[of ?x k, OF $\langle ?x \neq 1 \rangle$] by *auto*
 then have *sum*: *unity-root-sum* $k n = (?x^k - 1) / (?x - 1)$
 using *unity-root-sum-def unity-root-pow* by *simp*
 have $?x^k = 1$
 using *gr unity-root-eq-1-iff-int unity-root-pow* by *simp*
 then show *unity-root-sum* $k n = 0$ using *sum* by *auto*

qed

corollary *unity-root-sum-periodic-arithmetic:*

periodic-arithmetic (unity-root-sum k) k

unfolding *periodic-arithmetic-def*

proof

fix *n*

show *unity-root-sum k (n + k) = unity-root-sum k n*

by (*cases k = 0; cases k dvd n*) (*auto simp add: unity-root-sum*)

qed

lemma *unity-root-sum-nonzero-iff:*

fixes *r :: int*

assumes *k ≥ 1 and r ∈ {-k<..*

shows *unity-root-sum k r ≠ 0 ↔ r = 0*

proof

assume *unity-root-sum k r ≠ 0*

then have *k dvd r* **using** *unity-root-sum assms* **by** *blast*

then show *r = 0* **using** *assms(2)*

using *dvd-imp-le-int* **by** *force*

next

assume *r = 0*

then have *k dvd r* **by** *auto*

then have *unity-root-sum k r = k*

using *assms(1) unity-root-sum* **by** *blast*

then show *unity-root-sum k r ≠ 0* **using** *assms(1)* **by** *simp*

qed

end

5 Finite Fourier series

theory *Finite-Fourier-Series*

imports

Polynomial-Interpolation.Lagrange-Interpolation

Complex-Roots-Of-Unity

begin

5.1 Auxiliary facts

lemma *lagrange-exists:*

assumes *d: distinct (map fst zs-ws)*

defines *e: (p :: complex poly) ≡ lagrange-interpolation-poly zs-ws*

shows *degree p ≤ (length zs-ws) - 1*

(∀ x y. (x,y) ∈ set zs-ws → poly p x = y)

proof -

from *e* **show** *degree p ≤ (length zs-ws - 1)*

using *degree-lagrange-interpolation-poly* **by** *auto*

from *e d* **have**

```

    poly p x = y if (x,y) ∈ set zs-ws for x y
    using that lagrange-interpolation-poly by auto
    then show (∀ x y. (x,y) ∈ set zs-ws → poly p x = y)
    by auto
qed

lemma lagrange-unique:
  assumes o: length zs-ws > 0
  assumes d: distinct (map fst zs-ws)
  assumes 1: degree (p1 :: complex poly) ≤ (length zs-ws)-1 ∧
    (∀ x y. (x,y) ∈ set zs-ws → poly p1 x = y)
  assumes 2: degree (p2 :: complex poly) ≤ (length zs-ws)-1 ∧
    (∀ x y. (x,y) ∈ set zs-ws → poly p2 x = y)
  shows p1 = p2
proof (cases p1 - p2 = 0)
  case True then show ?thesis by simp
next
  case False
    have poly (p1-p2) x = 0 if x ∈ set (map fst zs-ws) for x
      using 1 2 that by (auto simp add: field-simps)
    from this d have 3: card {x. poly (p1-p2) x = 0} ≥ length zs-ws
    proof (induction zs-ws)
      case Nil then show ?case by simp
    next
      case (Cons z-w zs-ws)
        from False poly-roots-finite
        have f: finite {x. poly (p1 - p2) x = 0} by blast
        from Cons have set (map fst (z-w # zs-ws)) ⊆ {x. poly (p1 - p2) x = 0}
          by auto
        then have i: card (set (map fst (z-w # zs-ws))) ≤ card {x. poly (p1 - p2) x
= 0}
          using card-mono f by blast
        have length (z-w # zs-ws) ≤ card (set (map fst (z-w # zs-ws)))
          using Cons.prem2 distinct-card by fastforce
        from this i show ?case by simp
      qed
    from 1 2 have 4: degree (p1 - p2) ≤ (length zs-ws)-1
      using degree-diff-le by blast

  have p1 - p2 = 0
proof (rule ccontr)
  assume p1 - p2 ≠ 0
  then have card {x. poly (p1-p2) x = 0} ≤ degree (p1-p2)
    using poly-roots-degree by blast
  then have card {x. poly (p1-p2) x = 0} ≤ (length zs-ws)-1
    using 4 by auto
  then show False using 3 o by linarith
qed
then show ?thesis by simp

```


qed

Theorem 8.2

corollary *lagrange*:

assumes $\text{length } zs\text{-}ws > 0$ *distinct* ($\text{map } fst \text{ } zs\text{-}ws$)
shows $(\exists! (p :: \text{complex poly}).$
 $\text{degree } p \leq \text{length } zs\text{-}ws - 1 \wedge$
 $(\forall x y. (x, y) \in \text{set } zs\text{-}ws \longrightarrow \text{poly } p \ x = y))$
using *assms lagrange-exists lagrange-unique* **by** *blast*

lemma *poly-altdef'*:

assumes *gr*: $k \geq \text{degree } p$
shows $\text{poly } p \ (z :: \text{complex}) = (\sum_{i \leq k}. \text{coeff } p \ i * z^i)$
proof –
 {**fix** z
 have $1: \text{poly } p \ z = (\sum_{i \leq \text{degree } p}. \text{coeff } p \ i * z^i)$
 using *poly-altdef[of p z]* **by** *simp*
 have $\text{poly } p \ z = (\sum_{i \leq k}. \text{coeff } p \ i * z^i)$
 using *gr*
 proof (*induction k*)
 case 0 **then show** *?case* **by** (*simp add: poly-altdef*)
 next
 case (*Suc k*)
 then show *?case*
 using 1 *le-degree not-less-eq-eq* **by** *fastforce*
 qed}
then show *?thesis* **using** *gr* **by** *blast*
qed

5.2 Definition and uniqueness

definition *finite-fourier-poly* :: *complex list* \Rightarrow *complex poly* **where**

finite-fourier-poly $ws =$
 ($\text{let } k = \text{length } ws$
 in *poly-of-list* $[1 / k * (\sum_{m < k}. ws ! m * \text{unity-root } k \ (-n * m)). n \leftarrow [0..<k]]$)

lemma *degree-poly-of-list-le*: $\text{degree } (\text{poly-of-list } ws) \leq \text{length } ws - 1$

by (*intro degree-le*) (*auto simp: nth-default-def*)

lemma *degree-finite-fourier-poly*: $\text{degree } (\text{finite-fourier-poly } ws) \leq \text{length } ws - 1$

unfolding *finite-fourier-poly-def*

proof (*subst Let-def*)

let *?unrolled-list* =
 ($\text{map } (\lambda n. \text{complex-of-real } (1 / \text{real } (\text{length } ws)) * (\sum_{m < \text{length } ws}. ws ! m * \text{unity-root } (\text{length } ws) \ (- \text{int } n * \text{int } m)))$
 $[0..<\text{length } ws]$)

have $\text{degree } (\text{poly-of-list } ?unrolled\text{-list}) \leq \text{length } ?unrolled\text{-list} - 1$

by (rule degree-poly-of-list-le)
 also have ... = length [0..
 using length-map by auto
 also have ... = length ws - 1 by auto
 finally show degree (poly-of-list ?unrolled-list) ≤ length ws - 1 by blast
 qed

lemma *coeff-finite-fourier-poly*:
 assumes $n < \text{length } ws$
 defines $k \equiv \text{length } ws$
 shows $\text{coeff } (\text{finite-fourier-poly } ws) n =$
 $(1/k) * (\sum m < k. ws ! m * \text{unity-root } k (-n*m))$
 using *assms degree-finite-fourier-poly*
 by (auto simp: Let-def nth-default-def finite-fourier-poly-def)

lemma *poly-finite-fourier-poly*:
 fixes $m :: \text{int}$ and ws
 defines $k \equiv \text{length } ws$
 assumes $m \in \{0..<k\}$
 assumes $m < \text{length } ws$
 shows $\text{poly } (\text{finite-fourier-poly } ws) (\text{unity-root } k m) = ws ! (nat m)$
proof –
 have $k > 0$ using *assms* by auto

have *distr*:
 $(\sum j < \text{length } ws. ws ! j * \text{unity-root } k (-i*j)) * (\text{unity-root } k (m*i)) =$
 $(\sum j < \text{length } ws. ws ! j * \text{unity-root } k (-i*j) * (\text{unity-root } k (m*i)))$
 for i
 using *sum-distrib-right*[of $\lambda j. ws ! j * \text{unity-root } k (-i*j)$
 $\{..<k\} (\text{unity-root } k (m*i))$]
 using *k-def* by blast

{fix $j i :: \text{nat}$
 have $\text{unity-root } k (-i*j) * (\text{unity-root } k (m*i)) = \text{unity-root } k (-i*j + m*i)$
 by (*simp add: unity-root-diff unity-root-uminus field-simps*)
 also have ... = $\text{unity-root } k (i*(m-j))$
 by (*simp add: algebra-simps*)
 finally have $\text{unity-root } k (-i*j) * (\text{unity-root } k (m*i)) = \text{unity-root } k (i*(m-j))$
 by *simp*
 then have $ws ! j * \text{unity-root } k (-i*j) * (\text{unity-root } k (m*i)) =$
 $ws ! j * \text{unity-root } k (i*(m-j))$
 by *auto*
 } **note** *prod = this*

have *zeros*:
 $(\text{unity-root-sum } k (m-j) \neq 0 \longleftrightarrow m = j)$
 if $j \geq 0 \wedge j < k$ for j
 using *k-def that assms unity-root-sum-nonzero-iff*[of $- m - j$] by *simp*
 then have *sum-eq*:

```

( $\sum_{j \leq k-1} ws ! j * \text{unity-root-sum } k (m-j)$ ) =
  ( $\sum_{j \in \{\text{nat } m\}} ws ! j * \text{unity-root-sum } k (m-j)$ )
using assms(2) by (intro sum.mono-neutral-right, auto)

have poly (finite-fourier-poly ws) (unity-root k m) =
  ( $\sum_{i \leq k-1} \text{coeff } (finite-fourier-poly ws) i * (\text{unity-root } k m) ^ i$ )
using degree-finite-fourier-poly[of ws] k-def
  poly-altdef'[of finite-fourier-poly ws k-1 unity-root k m] by blast
also have ... = ( $\sum_{i < k} \text{coeff } (finite-fourier-poly ws) i * (\text{unity-root } k m) ^ i$ )
using assms(2) by (intro sum.cong) auto
also have ... = ( $\sum_{i < k} 1 / k *$ 
  ( $\sum_{j < k} ws ! j * \text{unity-root } k (-i*j)$ ) * ( $\text{unity-root } k m$ ) ^  $i$ )
using coeff-finite-fourier-poly[of - ws] k-def by auto
also have ... = ( $\sum_{i < k} 1 / k *$ 
  ( $\sum_{j < k} ws ! j * \text{unity-root } k (-i*j) * (\text{unity-root } k (m*i))$ )
using unity-root-pow by auto
also have ... = ( $\sum_{i < k} 1 / k *$ 
  ( $\sum_{j < k} ws ! j * \text{unity-root } k (-i*j) * (\text{unity-root } k (m*i))$ )
using distr k-def by simp
also have ... = ( $\sum_{i < k} 1 / k *$ 
  ( $\sum_{j < k} ws ! j * \text{unity-root } k (i*(m-j))$ )
using prod by presburger
also have ... =  $1 / k * (\sum_{i < k} 1 / k *$ 
  ( $\sum_{j < k} ws ! j * \text{unity-root } k (i*(m-j))$ )
by (simp add: sum-distrib-left)
also have ... =  $1 / k * (\sum_{j < k} 1 / k *$ 
  ( $\sum_{i < k} ws ! j * \text{unity-root } k (i*(m-j))$ )
using sum.swap by fastforce
also have ... =  $1 / k * (\sum_{j < k} ws ! j * (\sum_{i < k} \text{unity-root } k (i*(m-j))))$ 
by (simp add: vector-space-over-itself.scale-sum-right)
also have ... =  $1 / k * (\sum_{j < k} ws ! j * \text{unity-root-sum } k (m-j))$ 
unfolding unity-root-sum-def by (simp add: algebra-simps)
also have ( $\sum_{j < k} ws ! j * \text{unity-root-sum } k (m-j)$ ) = ( $\sum_{j \leq k-1} ws ! j * \text{unity-root-sum } k (m-j)$ )
using ( $k > 0$ ) by (intro sum.cong) auto
also have ... = ( $\sum_{j \in \{\text{nat } m\}} ws ! j * \text{unity-root-sum } k (m-j)$ )
using sum-eq .
also have ... =  $ws ! (\text{nat } m) * k$ 
using assms(2) by (auto simp: algebra-simps)
finally have poly (finite-fourier-poly ws) (unity-root k m) =  $ws ! (\text{nat } m)$ 
using assms(2) by auto
then show ?thesis by simp
qed

```

Theorem 8.3

theorem *finite-fourier-poly-unique*:
assumes *length ws > 0*
defines $k \equiv \text{length } ws$
assumes ($\text{degree } p \leq k - 1$)

assumes $(\forall m \leq k-1. (ws ! m) = poly\ p\ (unity-root\ k\ m))$
shows $p = finite-fourier-poly\ ws$
proof –
let $?z = map\ (\lambda m. unity-root\ k\ m)\ [0..<k]$
have $k: k > 0$ **using** *assms* **by** *auto*
from k **have** $d1: distinct\ ?z$
unfolding *distinct-conv-nth* **using** *unity-root-eqD[OF k]* **by** *force*
let $?zs-ws = zip\ ?z\ ws$
from $d1\ k-def$ **have** $d2: distinct\ (map\ fst\ ?zs-ws)$ **by** *simp*
have $l2: length\ ?zs-ws > 0$ **using** *assms(1) k-def* **by** *auto*
have $l3: length\ ?zs-ws = k$ **by** (*simp add: k-def*)

from *degree-finite-fourier-poly* **have** $degree: degree\ (finite-fourier-poly\ ws) \leq k -$
 1
using *k-def* **by** *simp*

have *interp: poly (finite-fourier-poly ws) x = y*
if $(x, y) \in set\ ?zs-ws$ **for** $x\ y$
proof –
from *that* **obtain** n **where**
 $x = map\ (unity-root\ k\ \circ\ int)\ [0..<k] ! n \wedge$
 $y = ws ! n \wedge$
 $n < length\ ws$
using *in-set-zip[of (x,y) (map (unity-root k) (map int [0..<k])) ws]*
by *auto*
then **have**
 $x = unity-root\ k\ (int\ n) \wedge$
 $y = ws ! n \wedge$
 $n < length\ ws$
using *nth-map[of n [0..<k] unity-root k \circ int] k-def* **by** *simp*
thus *poly (finite-fourier-poly ws) x = y*
by (*simp add: poly-finite-fourier-poly k-def*)
qed

have *interp-p: poly p x = y if (x,y) \in set ?zs-ws for x y*
proof –
from *that* **obtain** n **where**
 $x = map\ (unity-root\ k\ \circ\ int)\ [0..<k] ! n \wedge$
 $y = ws ! n \wedge$
 $n < length\ ws$
using *in-set-zip[of (x,y) (map (unity-root k) (map int [0..<k])) ws]*
by *auto*
then **have** *rw: x = unity-root k (int n) y = ws ! n n < length ws*
using *nth-map[of n [0..<k] unity-root k \circ int] k-def* **by** *simp+*
show *poly p x = y*
unfolding *rw(1,2)* **using** *assms(4) rw(3) k-def* **by** *simp*
qed

from *lagrange-unique[of - p finite-fourier-poly ws] d2 l2*

have l :
 $\text{degree } p \leq k - 1 \wedge$
 $(\forall x y. (x, y) \in \text{set } ?zs\text{-}ws \longrightarrow \text{poly } p \ x = y) \implies$
 $\text{degree } (\text{finite-fourier-poly } ws) \leq k - 1 \wedge$
 $(\forall x y. (x, y) \in \text{set } ?zs\text{-}ws \longrightarrow \text{poly } (\text{finite-fourier-poly } ws) \ x = y) \implies$
 $p = (\text{finite-fourier-poly } ws)$
using $l3$ **by** *fastforce*
from *assms degree interp interp-p l3*
show $p = (\text{finite-fourier-poly } ws)$ **using** l **by** *blast*
qed

The following alternative formulation returns a coefficient

definition *finite-fourier-poly'* :: $(\text{nat} \Rightarrow \text{complex}) \Rightarrow \text{nat} \Rightarrow \text{complex poly}$ **where**
 $\text{finite-fourier-poly}' \ ws \ k =$
 $(\text{poly-of-list } [1 / k * (\sum m < k. (ws \ m) * \text{unity-root } k \ (-n * m)). \ n \leftarrow [0..<k]])$

lemma *finite-fourier-poly'-conv-finite-fourier-poly*:
 $\text{finite-fourier-poly}' \ ws \ k = \text{finite-fourier-poly} \ [ws \ n. \ n \leftarrow [0..<k]]$
unfolding *finite-fourier-poly-def finite-fourier-poly'-def* **by** *simp*

lemma *coeff-finite-fourier-poly'*:
assumes $n < k$
shows $\text{coeff } (\text{finite-fourier-poly}' \ ws \ k) \ n =$
 $(1/k) * (\sum m < k. (ws \ m) * \text{unity-root } k \ (-n * m))$
proof –
let $?ws = [ws \ n. \ n \leftarrow [0..<k]]$
have $\text{coeff } (\text{finite-fourier-poly}' \ ws \ k) \ n =$
 $\text{coeff } (\text{finite-fourier-poly} \ ?ws) \ n$
by (*simp add: finite-fourier-poly'-conv-finite-fourier-poly*)
also have $\text{coeff } (\text{finite-fourier-poly} \ ?ws) \ n =$
 $1 / k * (\sum m < k. (?ws \ m) * \text{unity-root } k \ (-n * m))$
using *assms by (auto simp: coeff-finite-fourier-poly)*
also have $\dots = (1/k) * (\sum m < k. (ws \ m) * \text{unity-root } k \ (-n * m))$
using *assms by simp*
finally show $?thesis$ **by** *simp*
qed

lemma *degree-finite-fourier-poly'*: $\text{degree } (\text{finite-fourier-poly}' \ ws \ k) \leq k - 1$
using *degree-finite-fourier-poly[of [ws n. n ← [0..<k]]]*
by (*auto simp: finite-fourier-poly'-conv-finite-fourier-poly*)

lemma *poly-finite-fourier-poly'*:
fixes $m :: \text{int}$ **and** k
assumes $m \in \{0..<k\}$
shows $\text{poly } (\text{finite-fourier-poly}' \ ws \ k) \ (\text{unity-root } k \ m) = ws \ (\text{nat } m)$
using *assms poly-finite-fourier-poly[of m [ws n. n ← [0..<k]]]*
by (*auto simp: finite-fourier-poly'-conv-finite-fourier-poly poly-finite-fourier-poly*)

lemma *finite-fourier-poly'-unique*:

assumes $k > 0$
assumes $\text{degree } p \leq k - 1$
assumes $\forall m \leq k-1. \text{ ws } m = \text{poly } p (\text{unity-root } k \ m)$
shows $p = \text{finite-fourier-poly}' \ \text{ws } k$
proof –
let $?ws = [\text{ws } n. n \leftarrow [0..<k]]$
from *finite-fourier-poly-unique* **have** $p = \text{finite-fourier-poly } ?ws$ **using** *assms* **by**
simp
also have $\dots = \text{finite-fourier-poly}' \ \text{ws } k$
using *finite-fourier-poly'-conv-finite-fourier-poly ..*
finally show $p = \text{finite-fourier-poly}' \ \text{ws } k$ **by** *blast*
qed

lemma *fourier-unity-root*:

fixes $k :: \text{nat}$
assumes $k > 0$
shows $\text{poly } (\text{finite-fourier-poly}' \ f \ k) (\text{unity-root } k \ m) =$
 $(\sum n < k. 1/k * (\sum m < k. (f \ m) * \text{unity-root } k \ (-n * m)) * \text{unity-root } k \ (m * n))$
proof –
have $\text{poly } (\text{finite-fourier-poly}' \ f \ k) (\text{unity-root } k \ m) =$
 $(\sum n \leq k-1. \text{coeff } (\text{finite-fourier-poly}' \ f \ k) \ n * (\text{unity-root } k \ m)^{\wedge} n)$
using *poly-altdef'[of finite-fourier-poly' f k k-1 unity-root k m]*
degree-finite-fourier-poly'[of f k] **by** *simp*
also have $\dots = (\sum n \leq k-1. \text{coeff } (\text{finite-fourier-poly}' \ f \ k) \ n * (\text{unity-root } k$
 $(m * n)))$
using *unity-root-pow* **by** *simp*
also have $\dots = (\sum n < k. \text{coeff } (\text{finite-fourier-poly}' \ f \ k) \ n * (\text{unity-root } k \ (m * n)))$

using *assms* **by** (*intro sum.cong*) *auto*
also have $\dots = (\sum n < k. (1/k) * (\sum m < k. (f \ m) * \text{unity-root } k \ (-n * m)) * (\text{unity-root}$
 $k \ (m * n)))$
using *coeff-finite-fourier-poly'[of - k f]* **by** *simp*
finally show
 $\text{poly } (\text{finite-fourier-poly}' \ f \ k) (\text{unity-root } k \ m) =$
 $(\sum n < k. 1/k * (\sum m < k. (f \ m) * \text{unity-root } k \ (-n * m)) * \text{unity-root } k \ (m * n))$
by *blast*
qed

5.3 Expansion of an arithmetical function

Theorem 8.4

theorem *fourier-expansion-periodic-arithmetic*:

assumes $k > 0$
assumes *periodic-arithmetic f k*
defines $g \equiv (\lambda n. (1 / k) * (\sum m < k. f \ m * \text{unity-root } k \ (-n * m)))$
shows *periodic-arithmetic g k*
and $f \ m = (\sum n < k. g \ n * \text{unity-root } k \ (m * n))$
proof –
{fix l

```

from unity-periodic-arithmetic mult-period
have period: periodic-arithmetic ( $\lambda x. \text{unity-root } k \ x$ ) ( $k * l$ ) by simp}
note period = this
{fix  $n \ l$ 
have  $\text{unity-root } k \ (-(n+k)*l) = \text{cnj } (\text{unity-root } k \ ((n+k)*l))$ 
  by (simp add: unity-root-uminus unity-root-diff ring-distrib unity-root-add)
also have  $\text{unity-root } k \ ((n+k)*l) = \text{unity-root } k \ (n*l)$ 
  by (intro unity-root-cong) (auto simp: cong-def algebra-simps)
also have  $\text{cnj } \dots = \text{unity-root } k \ (-n*l)$ 
  using unity-root-uminus by simp
finally have  $\text{unity-root } k \ (-(n+k)*l) = \text{unity-root } k \ (-n*l)$  by simp}
note u-period = this

show 1: periodic-arithmetic g k
  unfolding periodic-arithmetic-def
proof
  fix  $n$ 

  have  $g(n+k) = (1 / k) * (\sum m < k. f(m) * \text{unity-root } k \ (-(n+k)*m))$ 
    using assms(3) by fastforce
  also have  $\dots = (1 / k) * (\sum m < k. f(m) * \text{unity-root } k \ (-n*m))$ 
  proof -
    have  $(\sum m < k. f(m) * \text{unity-root } k \ (-(n+k)*m)) =$ 
       $(\sum m < k. f(m) * \text{unity-root } k \ (-n*m))$ 
      by (intro sum.cong) (use u-period in auto)
    then show ?thesis by arg0
  qed
  also have  $\dots = g(n)$ 
    using assms(3) by fastforce
  finally show  $g(n+k) = g(n)$  by simp
qed

show  $f(m) = (\sum n < k. g(n) * \text{unity-root } k \ (m * \text{int } n))$ 
proof -
  {
    fix  $m$ 
    assume range: m ∈ {0..<k}
    have  $f(m) = (\sum n < k. g(n) * \text{unity-root } k \ (m * \text{int } n))$ 
    proof -
      have  $f \ m = \text{poly } (\text{finite-fourier-poly}' \ f \ k) \ (\text{unity-root } k \ m)$ 
        using range by (simp add: poly-finite-fourier-poly')
      also have  $\dots = (\sum n < k. (1 / k) * (\sum m < k. f(m) * \text{unity-root } k \ (-n*m)) *$ 
unity-root k (m*n))
        using fourier-unity-root assms(1) by blast
      also have  $\dots = (\sum n < k. g(n) * \text{unity-root } k \ (m*n))$ 
        using assms by simp
      finally show ?thesis by auto
    }
  qed
note concentrated = this

```

have *periodic-arithmetic* $(\lambda m. (\sum n < k. g(n) * \text{unity-root } k (m * \text{int } n))) k$
proof –
have *periodic-arithmetic* $(\lambda n. g(n) * \text{unity-root } k (i * \text{int } n)) k$ **for** $i :: \text{int}$
using 1 *unity-periodic-arithmetic mult-periodic-arithmetic*
unity-periodic-arithmetic-mult **by** *auto*
then have p -s: $\forall i < k. \text{periodic-arithmetic } (\lambda n. g(n) * \text{unity-root } k (i * \text{int } n)) k$
by *simp*
have *periodic-arithmetic* $(\lambda i. \sum n < k. g(n) * \text{unity-root } k (i * \text{int } n)) k$
unfolding *periodic-arithmetic-def*
proof
fix n
show $(\sum na < k. g \text{ na } * \text{unity-root } k (\text{int } (n + k) * \text{int } na)) =$
 $(\sum na < k. g \text{ na } * \text{unity-root } k (\text{int } n * \text{int } na))$
by (*intro sum.cong refl, simp add: distrib-right flip: of-nat-mult of-nat-add*)
(insert period, unfold periodic-arithmetic-def, blast)
qed
then show *?thesis* **by** *simp*
qed

from *this assms(1–2) concentrated*
unique-periodic-arithmetic-extension[of $k f (\lambda i. \sum n < k. g(n) * \text{unity-root } k (i * \text{int } n)) m$]
show $f m = (\sum n < k. g n * \text{unity-root } k (\text{int } m * \text{int } n))$ **by** *simp*
qed
qed

theorem *fourier-expansion-periodic-arithmetic-unique*:
fixes $f g :: \text{nat} \Rightarrow \text{complex}$
assumes $k > 0$
assumes *periodic-arithmetic* $f k$ **and** *periodic-arithmetic* $g k$
assumes $\bigwedge m. m < k \implies f m = (\sum n < k. g n * \text{unity-root } k (\text{int } (m * n)))$
shows $g n = (1 / k) * (\sum m < k. f m * \text{unity-root } k (-n * m))$
proof –
let $?p = \text{poly-of-list } [g(n). n \leftarrow [0..<k]]$
have d : *degree* $?p \leq k - 1$
proof –
have *degree* $?p \leq \text{length } [g(n). n \leftarrow [0..<k]] - 1$
using *degree-poly-of-list-le* **by** *blast*
also have $\dots = \text{length } [0..<k] - 1$
using *length-map* **by** *auto*
finally show *?thesis* **by** *simp*
qed
have c : *coeff* $?p i = (\text{if } i < k \text{ then } g(i) \text{ else } 0)$ **for** i
by (*simp add: nth-default-def*)
{fix z
have *poly* $?p z = (\sum n \leq k - 1. \text{coeff } ?p n * z^n)$
using *poly-altdef'*[of $?p k - 1$] d **by** *blast*
also have $\dots = (\sum n < k. \text{coeff } ?p n * z^n)$


```

    using <k > 0> by (intro sum.cong) auto
  also have ... = (∑ n<k. (if n < k then g(n) else 0) * z ^ n)
    using c by simp
  also have ... = (∑ n<k. g(n) * z ^ n)
    by (simp split: if-splits)
  finally have poly ?p z = (∑ n<k. g n * z ^ n) .}
  note eval = this
  {fix i
  have poly ?p (unity-root k i) = (∑ n<k. g(n) * (unity-root k i) ^ n)
    using eval by blast
  then have poly ?p (unity-root k i) = (∑ n<k. g(n) * (unity-root k (i*n)))
    using unity-root-pow by auto}
  note interpolation = this

```

```

{
  fix m
  assume b: m ≤ k-1
  from d assms(1)
  have f m = (∑ n<k. g(n) * unity-root k (m*n))
    using assms(4) b by auto
  also have ... = poly ?p (unity-root k m)
    using interpolation by simp
  finally have f m = poly ?p (unity-root k m) by auto
}

```

```

from this finite-fourier-poly'-unique[of k - f]
have p-is-fourier: ?p = finite-fourier-poly' f k
  using assms(1) d by blast

```

```

{
  fix n
  assume b: n ≤ k-1
  have f-1: coeff ?p n = (1 / k) * (∑ m<k. f(m) * unity-root k (-n*m))
    using p-is-fourier using assms(1) b by (auto simp: coeff-finite-fourier-poly')
  then have g(n) = (1 / k) * (∑ m<k. f(m) * unity-root k (-n*m))
    using c b assms(1)
  proof -
    have 1: coeff ?p n = (1 / k) * (∑ m<k. f(m) * unity-root k (-n*m))
      using f-1 by blast
    have 2: coeff ?p n = g n
      using c assms(1) b by simp
    show ?thesis using 1 2 by argo
  qed
}

```

```

have periodic-arithmetic (λn. (1 / k) * (∑ m<k. f(m) * unity-root k (-n*m)))
k
proof -

```

```

have periodic-arithmetic ( $\lambda i.$  unity-root  $k$  ( $-int\ i * int\ m$ ))  $k$  for  $m$ 
using unity-root-periodic-arithmetic-mult-minus by simp
then have periodic-arithmetic ( $\lambda i.$   $f(m) * unity-root\ k$  ( $-i * m$ ))  $k$  for  $m$ 
by (simp add: periodic-arithmetic-def)
then show periodic-arithmetic ( $\lambda i.$   $(1 / k) * (\sum m < k. f\ m * unity-root\ k$ 
( $-i * m$ )))  $k$ 
by (intro scalar-mult-periodic-arithmetic-fin-sum-periodic-arithmetic-set) auto
qed
note periodich = this
let ?h = ( $\lambda i.$   $(1 / k) * (\sum m < k. f\ m * unity-root\ k$  ( $-i * m$ )))
from unique-periodic-arithmetic-extension[ $of\ k\ g\ ?h\ n$ ]
  assms(3) assms(1) periodich
have  $g\ n = (1/k) * (\sum m < k. f\ m * unity-root\ k$  ( $-n * m$ ))
by (simp add: ( $\bigwedge na. na \leq k - 1 \implies g\ na = complex-of-real\ (1 / real\ k) *$ 
( $\sum m < k. f\ m * unity-root\ k$  ( $-int\ na * int\ m$ ))))
then show ?thesis by simp
qed
end

```

6 Ramanujan sums

```

theory Ramanujan-Sums
imports
  Dirichlet-Series.Moebius-Mu
  Gauss-Sums-Auxiliary
  Finite-Fourier-Series
begin

```

6.1 Basic sums

```

definition ramanujan-sum ::  $nat \Rightarrow nat \Rightarrow complex$ 
where ramanujan-sum  $k\ n = (\sum m \mid m \in \{1..k\} \wedge coprime\ m\ k. unity-root\ k$ 
( $m * n$ ))

```

```

notation ramanujan-sum ( $c$ )

```

```

lemma ramanujan-sum-0-n [simp]:  $c\ 0\ n = 0$ 
unfolding ramanujan-sum-def by simp

```

```

lemma sum-coprime-conv-dirichlet-prod-moebius-mu:
fixes  $F\ S :: nat \Rightarrow complex$  and  $f :: nat \Rightarrow nat \Rightarrow complex$ 
defines  $F \equiv (\lambda n. (\sum k \in \{1..n\}. f\ k\ n))$ 
defines  $S \equiv (\lambda n. (\sum k \mid k \in \{1..n\} \wedge coprime\ k\ n. f\ k\ n))$ 
assumes  $\bigwedge a\ b\ d. d\ dvd\ a \implies d\ dvd\ b \implies f\ (a\ div\ d)\ (b\ div\ d) = f\ a\ b$ 
shows  $S\ n = dirichlet-prod\ moebius-mu\ F\ n$ 
proof (cases  $n = 0$ )
case True
then show ?thesis

```

```

    using assms(2) unfolding dirichlet-prod-def by fastforce
next
case False
have  $S(n) = (\sum k \mid k \in \{1..n\} \wedge \text{coprime } k \ n . (f \ k \ n))$ 
  using assms by blast
also have ... =  $(\sum k \in \{1..n\}. (f \ k \ n) * \text{dirichlet-prod-neutral } (\text{gcd } k \ n))$ 
  using dirichlet-prod-neutral-intro by blast
also have ... =  $(\sum k \in \{1..n\}. (f \ k \ n) * (\sum d \mid d \ \text{dvd} \ (\text{gcd } k \ n). \text{moebius-mu } d))$ 
proof –
  {
    fix k
    have  $\text{dirichlet-prod-neutral } (\text{gcd } k \ n) = (\text{if } \text{gcd } k \ n = 1 \ \text{then } 1 \ \text{else } 0)$ 
      using dirichlet-prod-neutral-def[of gcd k n] by blast
    also have ... =  $(\sum d \mid d \ \text{dvd} \ \text{gcd } k \ n. \text{moebius-mu } d)$ 
      using sum-moebius-mu-divisors'[of gcd k n] by auto
    finally have  $\text{dirichlet-prod-neutral } (\text{gcd } k \ n) = (\sum d \mid d \ \text{dvd} \ \text{gcd } k \ n. \text{moebius-mu } d)$ 
      by auto
  } note summand = this
  then show ?thesis by (simp add: summand)
qed
also have ... =  $(\sum k = 1..n. (\sum d \mid d \ \text{dvd} \ \text{gcd } k \ n. (f \ k \ n) * \text{moebius-mu } d))$ 
  by (simp add: sum-distrib-left)
also have ... =  $(\sum k = 1..n. (\sum d \mid d \ \text{dvd} \ \text{gcd } n \ k. (f \ k \ n) * \text{moebius-mu } d))$ 
  using gcd.commute[of - n] by simp
also have ... =  $(\sum d \mid d \ \text{dvd} \ n. \sum k \mid k \in \{1..n\} \wedge d \ \text{dvd} \ k. (f \ k \ n) * \text{moebius-mu } d)$ 
  using sum.swap-restrict[of {1..n} {d. d dvd n}]
     $\lambda k \ d. (f \ k \ n) * \text{moebius-mu } d \ \lambda k \ d. d \ \text{dvd} \ k$ ] False by auto
also have ... =  $(\sum d \mid d \ \text{dvd} \ n. \text{moebius-mu } d * (\sum k \mid k \in \{1..n\} \wedge d \ \text{dvd} \ k. (f \ k \ n)))$ 
  by (simp add: sum-distrib-left mult.commute)
also have ... =  $(\sum d \mid d \ \text{dvd} \ n. \text{moebius-mu } d * (\sum q \in \{1..n \ \text{div} \ d\}. (f \ q \ (n \ \text{div} \ d))))$ 
proof –
  have st:
     $(\sum k \mid k \in \{1..n\} \wedge d \ \text{dvd} \ k. (f \ k \ n)) =$ 
       $(\sum q \in \{1..n \ \text{div} \ d\}. (f \ q \ (n \ \text{div} \ d)))$ 
    if  $d \ \text{dvd} \ n \ d > 0$  for  $d :: \text{nat}$ 
    by (rule sum.reindex-bij-witness[of - λk. k * d λk. k div d])
      (use assms(3) that in (fastforce simp: div-le-mono))+
  show ?thesis
    by (intro sum.cong) (use st False in fastforce)+
qed
also have ... =  $(\sum d \mid d \ \text{dvd} \ n. \text{moebius-mu } d * F(n \ \text{div} \ d))$ 
proof –
  have  $F(n \ \text{div} \ d) = (\sum q \in \{1..n \ \text{div} \ d\}. (f \ q \ (n \ \text{div} \ d)))$ 
    if  $d \ \text{dvd} \ n$  for  $d$ 
    by (simp add: F-def real-of-nat-div that)

```

then show ?thesis by auto
qed

also have ... = dirichlet-prod moebius-mu F n
by (simp add: dirichlet-prod-def)

finally show ?thesis by simp
qed

lemma dirichlet-prod-neutral-sum:

dirichlet-prod-neutral n = ($\sum k = 1..n. \text{unity-root } n \ k$) for n :: nat

proof (cases n = 0)

case True then show ?thesis unfolding dirichlet-prod-neutral-def by simp
next

case False

have 1: unity-root n 0 = 1 by simp

have 2: unity-root n n = 1

using unity-periodic-arithmetic[of n] add.left-neutral

proof -

have 1 = unity-root n (int 0)

using 1 by auto

also have unity-root n (int 0) = unity-root n (int (0 + n))

using unity-periodic-arithmetic[of n] periodic-arithmetic-def by algebra

also have ... = unity-root n (int n) by simp

finally show ?thesis by auto

qed

have ($\sum k = 1..n. \text{unity-root } n \ k$) = ($\sum k = 0..n. \text{unity-root } n \ k$) - 1

by (simp add: sum.atLeast-Suc-atMost sum.atLeast0-atMost-Suc-shift 1)

also have ... = (($\sum k = 0..n-1. \text{unity-root } n \ k$)+1) - 1

using sum.atLeast0-atMost-Suc[of ($\lambda k. \text{unity-root } n \ k$) n-1] False

by (simp add: 2)

also have ... = ($\sum k = 0..n-1. \text{unity-root } n \ k$)

by simp

also have ... = unity-root-sum n 1

unfolding unity-root-sum-def using (n ≠ 0) by (intro sum.cong) auto

also have ... = dirichlet-prod-neutral n

using unity-root-sum[of n 1] False

by (cases n = 1, auto simp add: False dirichlet-prod-neutral-def)

finally have 3: dirichlet-prod-neutral n = ($\sum k = 1..n. \text{unity-root } n \ k$) by auto

then show ?thesis by blast

qed

lemma moebius-coprime-sum:

moebius-mu n = ($\sum k \mid k \in \{1..n\} \wedge \text{coprime } k \ n. \text{unity-root } n \ (\text{int } k)$)

proof -

let ?f = ($\lambda k \ n. \text{unity-root } n \ k$)

from div-dvd-div have

d dvd a \implies d dvd b \implies

unity-root (a div d) (b div d) =

unity-root a b for a b d :: nat

using unity-root-def real-of-nat-div by fastforce

then have $(\sum k \mid k \in \{1..n\} \wedge \text{coprime } k \ n. \ ?f \ k \ n) =$
 $\text{dirichlet-prod moebius-mu } (\lambda n. \sum k = 1..n. \ ?f \ k \ n) \ n$
using *sum-coprime-conv-dirichlet-prod-moebius-mu*[of ?f n] **by blast**
also have $\dots = \text{dirichlet-prod moebius-mu dirichlet-prod-neutral } n$
by (*simp add: dirichlet-prod-neutral-sum*)
also have $\dots = \text{moebius-mu } n$
by (*cases n = 0*) (*simp-all add: dirichlet-prod-neutral-right-neutral*)
finally have $\text{moebius-mu } n = (\sum k \mid k \in \{1..n\} \wedge \text{coprime } k \ n. \ ?f \ k \ n)$
by *argo*
then show *?thesis* **by blast**
qed

corollary *ramanujan-sum-1-right* [*simp*]: $c \ k \ (\text{Suc } 0) = \text{moebius-mu } k$
unfolding *ramanujan-sum-def* **using** *moebius-coprime-sum*[of k] **by simp**

lemma *ramanujan-sum-dvd-eq-totient*:

assumes $k \ \text{dvd } n$
shows $c \ k \ n = \text{totient } k$
unfolding *ramanujan-sum-def*
proof –
have $\text{unity-root } k \ (m * n) = 1$ **for** m
using *assms* **by** (*cases k = 0*) (*auto simp: unity-root-eq-1-iff-int*)
then have $(\sum m \mid m \in \{1..k\} \wedge \text{coprime } m \ k. \ \text{unity-root } k \ (m * n)) =$
 $(\sum m \mid m \in \{1..k\} \wedge \text{coprime } m \ k. \ 1)$ **by** *simp*
also have $\dots = \text{card } \{m. \ m \in \{1..k\} \wedge \text{coprime } m \ k\}$ **by** *simp*
also have $\dots = \text{totient } k$
unfolding *totient-def totatives-def*
proof –
have $\{1..k\} = \{0 <..k\}$ **by** *auto*
then show $\text{of-nat } (\text{card } \{m \in \{1..k\}. \ \text{coprime } m \ k\}) =$
 $\text{of-nat } (\text{card } \{ka \in \{0 <..k\}. \ \text{coprime } ka \ k\})$ **by** *auto*
qed
finally show $(\sum m \mid m \in \{1..k\} \wedge \text{coprime } m \ k. \ \text{unity-root } k \ (m * n)) = \text{totient}$
 k
by *auto*
qed

6.2 Generalised sums

definition *gen-ramanujan-sum* :: $(\text{nat} \Rightarrow \text{complex}) \Rightarrow (\text{nat} \Rightarrow \text{complex}) \Rightarrow \text{nat} \Rightarrow$
 $\text{nat} \Rightarrow \text{complex}$ **where**

$$\text{gen-ramanujan-sum } f \ g = (\lambda k \ n. \sum d \mid d \ \text{dvd } \text{gcd } n \ k. \ f \ d * g \ (k \ \text{div } d))$$

notation *gen-ramanujan-sum* (s)

lemma *gen-ramanujan-sum-k-1*: $s \ f \ g \ k \ 1 = f \ 1 * g \ k$

unfolding *gen-ramanujan-sum-def* **by** *auto*

lemma *gen-ramanujan-sum-1-n*: $s \ f \ g \ 1 \ n = f \ 1 * g \ 1$

unfolding *gen-ramanujan-sum-def* **by** *simp*

lemma *gen-ramanujan-sum-periodic: periodic-arithmetic* (*s f g k*) *k*
unfolding *gen-ramanujan-sum-def periodic-arithmetic-def* **by** *simp*

Theorem 8.5

theorem *gen-ramanujan-sum-fourier-expansion:*

fixes *f g* :: *nat* \Rightarrow *complex* **and** *a* :: *nat* \Rightarrow *nat* \Rightarrow *complex*

assumes $k > 0$

defines $a \equiv (\lambda k m. (1/k) * (\sum d \mid d \text{ dvd } (gcd\ m\ k). g\ d * f\ (k\ \text{div}\ d) * d))$

shows $s\ f\ g\ k\ n = (\sum m \leq k-1. a\ k\ m * \text{unity-root}\ k\ (m*n))$

proof –

let $?g = (\lambda x. 1 / \text{of-nat}\ k * (\sum m < k. s\ f\ g\ k\ m * \text{unity-root}\ k\ (-x*m)))$

{fix *m* :: *nat*

let $?h = \lambda n\ d. f\ d * g\ (k\ \text{div}\ d) * \text{unity-root}\ k\ (-m * \text{int}\ n)$

have $(\sum l < k. s\ f\ g\ k\ l * \text{unity-root}\ k\ (-m*l)) =$

$(\sum l \in \{0..k-1\}. s\ f\ g\ k\ l * \text{unity-root}\ k\ (-m*l))$

using $\langle k > 0 \rangle$ **by** (*intro sum.cong*) *auto*

also have $\dots = (\sum l \in \{1..k\}. s\ f\ g\ k\ l * \text{unity-root}\ k\ (-m*l))$

proof –

have *periodic-arithmetic* $(\lambda l. \text{unity-root}\ k\ (-m*l))\ k$

using *unity-periodic-arithmetic-mult* **by** *blast*

then have *periodic-arithmetic* $(\lambda l. s\ f\ g\ k\ l * \text{unity-root}\ k\ (-m*l))\ k$

using *gen-ramanujan-sum-periodic mult-periodic-arithmetic* **by** *blast*

from *this periodic-arithmetic-sum-periodic-arithmetic-shift*[*of - k 1*]

have $\text{sum}\ (\lambda l. s\ f\ g\ k\ l * \text{unity-root}\ k\ (-m*l))\ \{0..k-1\} =$

$\text{sum}\ (\lambda l. s\ f\ g\ k\ l * \text{unity-root}\ k\ (-m*l))\ \{1..k\}$

using *assms(1) zero-less-one* **by** *simp*

then show *?thesis* **by** *argo*

qed

also have $\dots = (\sum n \in \{1..k\}. (\sum d \mid d \text{ dvd } (gcd\ n\ k). f(d) * g(k\ \text{div}\ d)) * \text{unity-root}\ k\ (-m*n))$

by (*simp add: gen-ramanujan-sum-def*)

also have $\dots = (\sum n \in \{1..k\}. (\sum d \mid d \text{ dvd } (gcd\ n\ k). f(d) * g(k\ \text{div}\ d) * \text{unity-root}\ k\ (-m*n)))$

by (*simp add: sum-distrib-right*)

also have $\dots = (\sum d \mid d \text{ dvd } k. \sum n \mid n \in \{1..k\} \wedge d \text{ dvd } n. ?h\ n\ d)$

proof –

have $(\sum n = 1..k. \sum d \mid d \text{ dvd } gcd\ n\ k. ?h\ n\ d) =$

$(\sum n = 1..k. \sum d \mid d \text{ dvd } k \wedge d \text{ dvd } n. ?h\ n\ d)$

using *gcd.commute*[*of - k*] **by** *simp*

also have $\dots = (\sum d \mid d \text{ dvd } k. \sum n \mid n \in \{1..k\} \wedge d \text{ dvd } n. ?h\ n\ d)$

using *sum.swap-restrict*[*of {1..k} {d. d dvd k}*

$- \lambda n\ d. d \text{ dvd } n]$ *assms* **by** *fastforce*

finally have

$(\sum n = 1..k. \sum d \mid d \text{ dvd } gcd\ n\ k. ?h\ n\ d) =$

$(\sum d \mid d \text{ dvd } k. \sum n \mid n \in \{1..k\} \wedge d \text{ dvd } n. ?h\ n\ d)$ **by** *blast*

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

qed

also have $\dots = (\sum d \mid d \text{ dvd } k. f(d)*g(k \text{ div } d)*$
 $(\sum n \mid n \in \{1..k\} \wedge d \text{ dvd } n. \text{unity-root } k (- m * \text{int } n)))$
by (*simp add: sum-distrib-left*)

also have $\dots = (\sum d \mid d \text{ dvd } k. f(d)*g(k \text{ div } d)*$
 $(\sum e \in \{1..k \text{ div } d\}. \text{unity-root } k (- m * (e*d))))$
using *assms(1) sum-div-reduce div-greater-zero-iff dvd-div-gt0* **by** *auto*

also have $\dots = (\sum d \mid d \text{ dvd } k. f(d)*g(k \text{ div } d)*$
 $(\sum e \in \{1..k \text{ div } d\}. \text{unity-root } (k \text{ div } d) (- m * e)))$

proof –

{
 fix $d \ e$
 assume $d \text{ dvd } k$
 hence $2 * \text{pi} * \text{real-of-int } (- \text{int } m * \text{int } (e * d)) / \text{real } k =$
 $2 * \text{pi} * \text{real-of-int } (- \text{int } m * \text{int } e) / \text{real } (k \text{ div } d)$ **by** *auto*
 hence $\text{unity-root } k (- m * (e * d)) = \text{unity-root } (k \text{ div } d) (- m * e)$
 unfolding *unity-root-def* **by** *simp*
 }

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

qed

also have $\dots = \text{dirichlet-prod } (\lambda d. f(d)*g(k \text{ div } d))$
 $(\lambda d. (\sum e \in \{1..d\}. \text{unity-root } d (- m * e))) \ k$
unfolding *dirichlet-prod-def* **by** *blast*

also have $\dots = \text{dirichlet-prod } (\lambda d. (\sum e \in \{1..d\}. \text{unity-root } d (- m * e)))$
 $(\lambda d. f(d)*g(k \text{ div } d)) \ k$
using *dirichlet-prod-commutes*[*of*
 $(\lambda d. f(d)*g(k \text{ div } d))$
 $(\lambda d. (\sum e \in \{1..d\}. \text{unity-root } d (- m * e)))]$ **by** *argo*

also have $\dots = (\sum d \mid d \text{ dvd } k.$
 $(\sum e \in \{1..(d::\text{nat})\}. \text{unity-root } d (- m * e))*f(k \text{ div } d)*g(k \text{ div } (k \text{ div } d)))$
unfolding *dirichlet-prod-def* **by** *blast*

also have $\dots = (\sum d \mid d \text{ dvd } k. (\sum e \in \{1..(d::\text{nat})\}.$
 $\text{unity-root } d (- m * e))*f(k \text{ div } d)*g(d))$

proof –

{
 fix $d :: \text{nat}$
 assume $d \text{ dvd } k$
 then have $k \text{ div } (k \text{ div } d) = d$
 by (*simp add: assms(1) div-div-eq-right*)
 }

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

qed

also have $\dots = (\sum (d::\text{nat}) \mid d \text{ dvd } k \wedge d \text{ dvd } m. d*(f(k \text{ div } d)*g(d)))$

proof –

{
 fix d
 assume $d \text{ dvd } k$
 with *assms* **have** $d > 0$ **by** (*intro Nat.gr0I*) *auto*
 have *periodic-arithmetic* $(\lambda x. \text{unity-root } d (- m * \text{int } x)) \ d$

```

    using unity-periodic-arithmetic-mult by blast
  then have  $(\sum e \in \{1..d\}. \text{unity-root } d (-m * e)) =$ 
     $(\sum e \in \{0..d-1\}. \text{unity-root } d (-m * e))$ 
    using periodic-arithmetic-sum-periodic-arithmetic-shift[of  $\lambda e. \text{unity-root } d$ 
 $(-m * e) \ d \ 1$ ] assms  $\langle d \ \text{dvd } k \rangle$ 
    by fastforce
  also have ... = unity-root-sum  $d (-m)$ 
    unfolding unity-root-sum-def using  $\langle d > 0 \rangle$  by (intro sum.cong) auto
  finally have
     $(\sum e \in \{1..d\}. \text{unity-root } d (-m * e)) = \text{unity-root-sum } d (-m)$ 
    by arg0
}
then have
 $(\sum d \mid d \ \text{dvd } k. (\sum e = 1..d. \text{unity-root } d (-m * \text{int } e)) * (f (k \ \text{div } d) * g \ d))$ 
=
 $(\sum d \mid d \ \text{dvd } k. \text{unity-root-sum } d (-m) * (f (k \ \text{div } d) * g \ d))$  by simp
also have ... =  $(\sum d \mid d \ \text{dvd } k \wedge d \ \text{dvd } m. \text{unity-root-sum } d (-m) * (f (k \ \text{div } d) * g \ d))$ 
proof (intro sum.mono-neutral-right, simp add:  $\langle k > 0 \rangle$ , blast, standard)
  fix  $i$ 
  assume as:  $i \in \{d. d \ \text{dvd } k\} - \{d. d \ \text{dvd } k \wedge d \ \text{dvd } m\}$ 
  then have  $i \geq 1$  using  $\langle k > 0 \rangle$  by auto
  have  $k \geq 1$  using  $\langle k > 0 \rangle$  by auto
  have  $\neg i \ \text{dvd } (-m)$  using as by auto
  thus unity-root-sum  $i (- \ \text{int } m) * (f (k \ \text{div } i) * g \ i) = 0$ 
    using  $\langle i \geq 1 \rangle$  by (subst unity-root-sum(2)) auto
qed
also have ... =  $(\sum d \mid d \ \text{dvd } k \wedge d \ \text{dvd } m. d * (f (k \ \text{div } d) * g \ d))$ 
proof -
  {fix  $d :: \text{nat}$ 
    assume 1:  $d \ \text{dvd } m$ 
    assume 2:  $d \ \text{dvd } k$ 
    then have unity-root-sum  $d (-m) = d$ 
      using unity-root-sum[of  $d (-m)$ ] assms(1) 1 2
      by auto}
  then show ?thesis by auto
qed
finally show ?thesis by arg0
qed
also have ... =  $(\sum d \mid d \ \text{dvd } \text{gcd } m \ k. \text{of-nat } d * (f (k \ \text{div } d) * g \ d))$ 
  by (simp add: gcd.commute)
also have ... =  $(\sum d \mid d \ \text{dvd } \text{gcd } m \ k. g \ d * f (k \ \text{div } d) * d)$ 
  by (simp add: algebra-simps sum-distrib-left)
also have  $1 / k * \dots = a \ k \ m$  using a-def by auto
finally have ?g m = a k m by simp}
note a-eq-g = this
{
  fix  $m$ 
  from fourier-expansion-periodic-arithmetic(2)[of  $k \ s \ f \ g \ k$ ] gen-ramanujan-sum-periodic

```



```

assms(1)
  have  $s f g k m = (\sum n < k. ?g n * \text{unity-root } k (int m * n))$ 
    by blast
  also have  $\dots = (\sum n < k. a k n * \text{unity-root } k (int m * n))$ 
    using a-eq-g by simp
  also have  $\dots = (\sum n \leq k-1. a k n * \text{unity-root } k (int m * n))$ 
    using  $\langle k > 0 \rangle$  by (intro sum.cong) auto
  finally have  $s f g k m =$ 
     $(\sum n \leq k-1. a k n * \text{unity-root } k (int n * int m))$ 
    by (simp add: algebra-simps)
}
then show ?thesis by blast
qed

```

Theorem 8.6

theorem *ramanujan-sum-dirichlet-form*:

```

  fixes  $k n :: nat$ 
  assumes  $k > 0$ 
  shows  $c k n = (\sum d \mid d \text{ dvd } gcd n k. d * \text{moebius-mu } (k \text{ div } d))$ 
proof -
  define  $a :: nat \Rightarrow nat \Rightarrow complex$ 
    where  $a = (\lambda k m.$ 
       $1 / \text{of-nat } k * (\sum d \mid d \text{ dvd } gcd m k. \text{moebius-mu } d * \text{of-nat } (k \text{ div } d) * \text{of-nat } d))$ 
    {fix  $m$ 
  have  $a k m = (\text{if } gcd m k = 1 \text{ then } 1 \text{ else } 0)$ 
  proof -
  have  $a k m = 1 / \text{of-nat } k * (\sum d \mid d \text{ dvd } gcd m k. \text{moebius-mu } d * \text{of-nat } (k \text{ div } d) * \text{of-nat } d)$ 
    unfolding a-def by blast
  also have 2:  $\dots = 1 / \text{of-nat } k * (\sum d \mid d \text{ dvd } gcd m k. \text{moebius-mu } d * \text{of-nat } (k \text{ div } d) * \text{of-nat } d)$ 
  proof -
    {fix  $d :: nat$ 
  assume dvd:  $d \text{ dvd } gcd m k$ 
  have  $\text{moebius-mu } d * \text{of-nat } (k \text{ div } d) * \text{of-nat } d = \text{moebius-mu } d * \text{of-nat } k$ 
  proof -
    have  $(k \text{ div } d) * d = k$  using dvd by auto
    then show  $\text{moebius-mu } d * \text{of-nat } (k \text{ div } d) * \text{of-nat } d = \text{moebius-mu } d * \text{of-nat } k$ 
      by (simp add: algebra-simps, subst of-nat-mult[symmetric], simp)
    }
  qed} note eq = this
  show ?thesis using sum.cong by (simp add: eq)
  }
qed

  also have 3:  $\dots = (\sum d \mid d \text{ dvd } gcd m k. \text{moebius-mu } d)$ 
    by (simp add: sum-distrib-left assms)
  also have 4:  $\dots = (\text{if } gcd m k = 1 \text{ then } 1 \text{ else } 0)$ 

```

```

    using sum-moebius-mu-divisors' by blast
  finally show a k m = (if gcd m k = 1 then 1 else 0)
    using coprime-def by blast
qed} note a-expr = this

let ?f = (λm. (if gcd m k = 1 then 1 else 0) *
          unity-root k (int m * n))
from gen-ramanujan-sum-fourier-expansion[of k id moebius-mu n] assms
have s (λx. of-nat (id x)) moebius-mu k n =
  (∑ m ≤ k - 1.
    1 / of-nat k *
    (∑ d | d dvd gcd m k.
      moebius-mu d * of-nat (k div d) * of-nat d) *
    unity-root k (int m * n)) by simp
also have ... = (∑ m ≤ k - 1.
  a k m *
  unity-root k (int m * n)) using a-def by blast
also have ... = (∑ m ≤ k - 1.
  (if gcd m k = 1 then 1 else 0) *
  unity-root k (int m * n)) using a-expr by auto
also have ... = (∑ m ∈ {1..k}.
  (if gcd m k = 1 then 1 else 0) *
  unity-root k (int m * n))
proof -
  have periodic-arithmetic (λm. (if gcd m k = 1 then 1 else 0) *
    unity-root k (int m * n)) k
  proof -
    have periodic-arithmetic (λm. if gcd m k = 1 then 1 else 0) k
      by (simp add: periodic-arithmetic-def)
    moreover have periodic-arithmetic (λm. unity-root k (int m * n)) k
      using unity-periodic-arithmetic-mult[of k n]
      by (subst mult.commute, simp)
    ultimately show periodic-arithmetic ?f k
      using mult-periodic-arithmetic by simp
  qed
  then have sum ?f {0..k - 1} = sum ?f {1..k}
    using periodic-arithmetic-sum-periodic-arithmetic-shift[of ?f k 1] by force
  then show ?thesis by (simp add: atMost-atLeast0)
qed
also have ... = (∑ m | m ∈ {1..k} ∧ gcd m k = 1.
  (if gcd m k = 1 then 1 else 0) *
  unity-root k (int m * int n))
  by (intro sum.mono-neutral-right, auto)
also have ... = (∑ m | m ∈ {1..k} ∧ gcd m k = 1.
  unity-root k (int m * int n)) by simp
also have ... = (∑ m | m ∈ {1..k} ∧ coprime m k.
  unity-root k (int m * int n))
  using coprime-iff-gcd-eq-1 by presburger
also have ... = c k n unfolding ramanujan-sum-def by simp

```

finally show *?thesis* **unfolding** *gen-ramanujan-sum-def* **by** *auto*
qed

corollary *ramanujan-sum-conv-gen-ramanujan-sum*:

$k > 0 \implies c\ k\ n = s\ id\ moebius\ mu\ k\ n$

using *ramanujan-sum-dirichlet-form* **unfolding** *gen-ramanujan-sum-def* **by** *simp*

Theorem 8.7

theorem *gen-ramanujan-sum-distrib*:

fixes $f\ g :: nat \Rightarrow complex$

assumes $a > 0\ b > 0\ m > 0\ k > 0$

assumes *coprime a k coprime b m coprime k m*

assumes *multiplicative-function f and*

multiplicative-function g

shows $s\ f\ g\ (m*k)\ (a*b) = s\ f\ g\ m\ a * s\ f\ g\ k\ b$

proof –

from *assms(1–6)* **have** $eq: gcd\ (m*k)\ (a*b) = gcd\ a\ m * gcd\ k\ b$

by (*simp add: linear-gcd gcd.commute mult.commute*)

have $s\ f\ g\ (m*k)\ (a*b) =$

$(\sum d \mid d\ dvd\ gcd\ (m*k)\ (a*b). f(d) * g((m*k)\ div\ d))$

unfolding *gen-ramanujan-sum-def* **by** (*rule sum.cong, simp add: gcd.commute,blast*)

also have $\dots =$

$(\sum d \mid d\ dvd\ gcd\ a\ m * gcd\ k\ b. f(d) * g((m*k)\ div\ d))$

using *eq* **by** *simp*

also have $\dots =$

$(\sum (d1,d2) \mid d1\ dvd\ gcd\ a\ m \wedge d2\ dvd\ gcd\ k\ b. f(d1*d2) * g((m*k)\ div\ (d1*d2)))$

proof –

have $b: bij\ betw\ (\lambda(d1, d2). d1 * d2)$

$\{(d1, d2). d1\ dvd\ gcd\ a\ m \wedge d2\ dvd\ gcd\ k\ b\}$

$\{d. d\ dvd\ gcd\ a\ m * gcd\ k\ b\}$

using *assms(5) reindex-product-bij* **by** *blast*

have $(\sum (d1, d2) \mid d1\ dvd\ gcd\ a\ m \wedge d2\ dvd\ gcd\ k\ b.$

$f\ (d1 * d2) * g\ (m * k\ div\ (d1 * d2))) =$

$(\sum x \in \{(d1, d2). d1\ dvd\ gcd\ a\ m \wedge d2\ dvd\ gcd\ k\ b\}.$

$f\ (case\ x\ of\ (d1, d2) \Rightarrow d1 * d2) *$

$g\ (m * k\ div\ (case\ x\ of\ (d1, d2) \Rightarrow d1 * d2)))$

by (*rule sum.cong,auto*)

also have $\dots = (\sum d \mid d\ dvd\ gcd\ a\ m * gcd\ k\ b. f\ d * g\ (m * k\ div\ d))$

using b **by** (*rule sum.reindex-bij-betw[of $\lambda(d1,d2). d1*d2$]*)

finally show *?thesis* **by** *argo*

qed

also have $\dots = (\sum d1 \mid d1\ dvd\ gcd\ a\ m. \sum d2 \mid d2\ dvd\ gcd\ k\ b.$

$f\ (d1*d2) * g\ ((m*k)\ div\ (d1*d2)))$

by (*simp add: sum.cartesian-product*) (*rule sum.cong,auto*)

also have $\dots = (\sum d1 \mid d1\ dvd\ gcd\ a\ m. \sum d2 \mid d2\ dvd\ gcd\ k\ b.$

$f\ d1 * f\ d2 * g\ ((m*k)\ div\ (d1*d2)))$

using *assms(5) assms(8) multiplicative-function.mult-coprime*

by (*intro sum.cong refl*) *fastforce+*
 also have ... = $(\sum d1 \mid d1 \text{ dvd gcd } a \ m. \sum d2 \mid d2 \text{ dvd gcd } k \ b. f \ d1 * f \ d2 * g \ (m \ \text{div} \ d1) * g \ (k \ \text{div} \ d2))$
proof (*intro sum.cong refl, clarify, goal-cases*)
 case (1 *d1 d2*)
 hence $g \ (m * k \ \text{div} \ (d1 * d2)) = g \ (m \ \text{div} \ d1) * g \ (k \ \text{div} \ d2)$
 using *assms(7,9) multipl-div*
 by (*meson coprime-commute dvd-gcdD1 dvd-gcdD2*)
 thus ?*case* by *simp*
qed
 also have ... = $(\sum i \in \{d1. \ d1 \ \text{dvd} \ \text{gcd} \ a \ m\}. \sum j \in \{d2. \ d2 \ \text{dvd} \ \text{gcd} \ k \ b\}. f \ i * g \ (m \ \text{div} \ i) * (f \ j * g \ (k \ \text{div} \ j)))$
 by (*rule sum.cong,blast,rule sum.cong,blast,simp*)
 also have ... = $(\sum d1 \mid d1 \ \text{dvd} \ \text{gcd} \ a \ m. f \ d1 * g \ (m \ \text{div} \ d1)) * (\sum d2 \mid d2 \ \text{dvd} \ \text{gcd} \ k \ b. f \ d2 * g \ (k \ \text{div} \ d2))$
 by (*simp add: sum-product*)
 also have ... = $s \ f \ g \ m \ a * s \ f \ g \ k \ b$
 unfolding *gen-ramanujan-sum-def* by (*simp add: gcd.commute*)
 finally show ?*thesis* by *blast*
qed

corollary *gen-ramanujan-sum-distrib-right:*

fixes $f \ g :: \text{nat} \Rightarrow \text{complex}$
 assumes $a > 0$ and $b > 0$ and $m > 0$
 assumes *coprime* $b \ m$
 assumes *multiplicative-function* f and
 multiplicative-function g
 shows $s \ f \ g \ m \ (a * b) = s \ f \ g \ m \ a$
proof –
 have $s \ f \ g \ m \ (a*b) = s \ f \ g \ m \ a * s \ f \ g \ 1 \ b$
 using *assms gen-ramanujan-sum-distrib[of a b m 1 f g]* by *simp*
 also have ... = $s \ f \ g \ m \ a * f \ 1 * g \ 1$
 using *gen-ramanujan-sum-1-n* by *auto*
 also have ... = $s \ f \ g \ m \ a$
 using *assms(5-6)*
 by (*simp add: multiplicative-function-def*)
 finally show $s \ f \ g \ m \ (a*b) = s \ f \ g \ m \ a$ by *blast*
qed

corollary *gen-ramanujan-sum-distrib-left:*

fixes $f \ g :: \text{nat} \Rightarrow \text{complex}$
 assumes $a > 0$ and $k > 0$ and $m > 0$
 assumes *coprime* $a \ k$ and *coprime* $k \ m$
 assumes *multiplicative-function* f and
 multiplicative-function g
 shows $s \ f \ g \ (m*k) \ a = s \ f \ g \ m \ a * g \ k$
proof –
 have $s \ f \ g \ (m*k) \ a = s \ f \ g \ m \ a * s \ f \ g \ k \ 1$
 using *assms gen-ramanujan-sum-distrib[of a 1 m k f g]* by *simp*

also have $\dots = s f g m a * f(1) * g(k)$
using *gen-ramanujan-sum-k-1* **by** *auto*
also have $\dots = s f g m a * g k$
using *assms(6)*
by (*simp add: multiplicative-function-def*)
finally show *?thesis* **by** *blast*
qed

corollary *ramanujan-sum-distrib:*

assumes $a > 0$ **and** $k > 0$ **and** $m > 0$ **and** $b > 0$
assumes *coprime a k coprime b m coprime m k*
shows $c (m*k) (a*b) = c m a * c k b$

proof –

have $c (m*k) (a*b) = s id moebius-mu (m*k) (a*b)$
using *ramanujan-sum-conv-gen-ramanujan-sum assms(2,3)* **by** *simp*

also have $\dots = (s id moebius-mu m a) * (s id moebius-mu k b)$
using *gen-ramanujan-sum-distrib[of a b m k id moebius-mu]*
assms mult-id mult-moebius mult-of-nat
coprime-commute[of m k] **by** *auto*

also have $\dots = c m a * c k b$ **using** *ramanujan-sum-conv-gen-ramanujan-sum*
assms **by** *simp*

finally show *?thesis* **by** *simp*

qed

corollary *ramanujan-sum-distrib-right:*

assumes $a > 0$ **and** $k > 0$ **and** $m > 0$ **and** $b > 0$
assumes *coprime b m*
shows $c m (a*b) = c m a$

using *assms ramanujan-sum-conv-gen-ramanujan-sum mult-id mult-moebius*
mult-of-nat gen-ramanujan-sum-distrib-right **by** *auto*

corollary *ramanujan-sum-distrib-left:*

assumes $a > 0$ $k > 0$ $m > 0$
assumes *coprime a k coprime m k*
shows $c (m*k) a = c m a * moebius-mu k$
using *assms*

by (*simp add: ramanujan-sum-conv-gen-ramanujan-sum, subst gen-ramanujan-sum-distrib-left*)
(auto simp: coprime-commute mult-of-nat mult-moebius)

lemma *dirichlet-prod-completely-multiplicative-left:*

fixes $f h :: nat \Rightarrow complex$ **and** $k :: nat$
defines $g \equiv (\lambda k. moebius-mu k * h k)$

defines $F \equiv dirichlet-prod f g$

assumes $k > 0$

assumes *completely-multiplicative-function f*
multiplicative-function h

assumes $\bigwedge p. prime p \implies f(p) \neq 0 \wedge f(p) \neq h(p)$

shows $F k = f k * (\prod_{p \in prime-factors k} 1 - h p / f p)$

proof –
have 1: *multiplicative-function* $(\lambda p. h(p) \text{ div } f(p))$
using *multiplicative-function-divide*
comp-to-mult assms(4,5) **by** *blast*
have $F\ k = \text{dirichlet-prod } g\ f\ k$
unfolding *F-def* **using** *dirichlet-prod-commutes[of f g]* **by** *auto*
also have $\dots = (\sum d \mid d \text{ dvd } k. \text{moebius-mu } d * h\ d * f(k \text{ div } d))$
unfolding *g-def* *dirichlet-prod-def* **by** *blast*
also have $\dots = (\sum d \mid d \text{ dvd } k. \text{moebius-mu } d * h\ d * (f(k) \text{ div } f(d)))$
using *multipl-div-mono[of f - k]* *assms(4,6)*
by *(intro sum.cong, auto, force)*
also have $\dots = f\ k * (\sum d \mid d \text{ dvd } k. \text{moebius-mu } d * (h\ d \text{ div } f(d)))$
by *(simp add: sum-distrib-left algebra-simps)*
also have $\dots = f\ k * (\prod_{p \in \text{prime-factors } k}. 1 - (h\ p \text{ div } f\ p))$
using *sum-divisors-moebius-mu-times-multiplicative[of $\lambda p. h\ p \text{ div } f\ p$]* *1*
assms(3) **by** *simp*
finally show *F-eq: $F\ k = f\ k * (\prod_{p \in \text{prime-factors } k}. 1 - (h\ p \text{ div } f\ p))$*
by *blast*
qed

Theorem 8.8

theorem *gen-ramanujan-sum-dirichlet-expr:*
fixes $f\ h :: \text{nat} \Rightarrow \text{complex}$ **and** $n\ k :: \text{nat}$
defines $g \equiv (\lambda k. \text{moebius-mu } k * h\ k)$
defines $F \equiv \text{dirichlet-prod } f\ g$
defines $N \equiv k \text{ div } \text{gcd } n\ k$
assumes *completely-multiplicative-function* f
multiplicative-function h
assumes $\bigwedge p. \text{prime } p \implies f(p) \neq 0 \wedge f(p) \neq h(p)$
assumes $k > 0\ n > 0$
shows $s\ f\ g\ k\ n = (F(k) * g(N)) \text{ div } (F(N))$

proof –
define a **where** $a \equiv \text{gcd } n\ k$
have 2: $k = a * N$ **unfolding** *a-def* *N-def* **by** *auto*
have 3: $a > 0$ **using** *a-def* *assms(7,8)* **by** *simp*
have *Ngr0: $N > 0$* **using** *assms(7,8)* *2 N-def* **by** *fastforce*
have *f-k-not-z: $f\ k \neq 0$*
using *completely-multiplicative-nonzero* *assms(4,6,7)* **by** *blast*
have *f-N-not-z: $f\ N \neq 0$*
using *completely-multiplicative-nonzero* *assms(4,6)* *Ngr0* **by** *blast*
have *bij: bij-betw* $(\lambda d. a \text{ div } d) \{d. d \text{ dvd } a\} \{d. d \text{ dvd } a\}$
unfolding *bij-betw-def*
proof
show *inj: inj-on* $(\lambda d. a \text{ div } d) \{d. d \text{ dvd } a\}$
using *inj-on-def* *3 dvd-div-eq-2* **by** *blast*
show *surj: $(\lambda d. a \text{ div } d) \{d. d \text{ dvd } a\} = \{d. d \text{ dvd } a\}$*
unfolding *image-def*
proof
show $\{y. \exists x \in \{d. d \text{ dvd } a\}. y = a \text{ div } x\} \subseteq \{d. d \text{ dvd } a\}$

```

    by auto
  show {d. d dvd a} ⊆ {y. ∃ x ∈ {d. d dvd a}. y = a div x}
  proof
    fix d
    assume a: d ∈ {d. d dvd a}
    from a have 1: (a div d) ∈ {d. d dvd a} by auto
    from a have 2: d = a div (a div d) using 3 by auto
    from 1 2 show d ∈ {y. ∃ x ∈ {d. d dvd a}. y = a div x} by blast
  qed
  qed
  qed

  have s f g k n = (∑ d | d dvd a. f(d)*moebius-mu(k div d)*h(k div d))
    unfolding gen-ramanujan-sum-def g-def a-def by (simp add: mult.assoc)
  also have ... = (∑ d | d dvd a. f(d) * moebius-mu(a*N div d)*h(a*N div d))
    using 2 by blast
  also have ... = (∑ d | d dvd a. f(a div d) * moebius-mu(N*d)*h(N*d))
    (is ?a = ?b)
  proof -
    define f-aux where f-aux ≡ (λd. f d * moebius-mu (a * N div d) * h (a * N
  div d))
    have 1: ?a = (∑ d | d dvd a. f-aux d) using f-aux-def by blast
    {fix d :: nat
     assume d dvd a
     then have N * a div (a div d) = N * d
       using 3 by force}
    then have 2: ?b = (∑ d | d dvd a. f-aux (a div d))
      unfolding f-aux-def by (simp add: algebra-simps)
    show ?a = ?b
      using bij 1 2
      by (simp add: sum.reindex-bij-betw[of ((div) a) {d. d dvd a} {d. d dvd a}])
  qed
  also have ... = moebius-mu N * h N * f a * (∑ d | d dvd a ∧ coprime N d.
  moebius-mu d * (h d div f d))
    (is ?a = ?b)
  proof -
    have ?a = (∑ d | d dvd a ∧ coprime N d. f(a div d) * moebius-mu (N*d) * h
  (N*d))
      by (rule sum.mono-neutral-right)(auto simp add: moebius-prod-not-coprime
  3)
    also have ... = (∑ d | d dvd a ∧ coprime N d. moebius-mu N * h N * f(a div
  d) * moebius-mu d * h d)
      proof (rule sum.cong,simp)
        fix d
        assume a: d ∈ {d. d dvd a ∧ coprime N d}
        then have 1: moebius-mu (N*d) = moebius-mu N * moebius-mu d
          using mult-moebius unfolding multiplicative-function-def
          by (simp add: moebius-mu.mult-coprime)
        from a have 2: h (N*d) = h N * h d

```

```

    using assms(5) unfolding multiplicative-function-def
    by (simp add: assms(5) multiplicative-function.mult-coprime)
  show  $f (a \operatorname{div} d) * \operatorname{moebius-mu} (N * d) * h (N * d) =$ 
     $\operatorname{moebius-mu} N * h N * f (a \operatorname{div} d) * \operatorname{moebius-mu} d * h d$ 
    by (simp add: divide-simps 1 2)
  qed
  also have ... =  $(\sum d \mid d \operatorname{dvd} a \wedge \operatorname{coprime} N d. \operatorname{moebius-mu} N * h N * (f a$ 
div f d) * moebius-mu d * h d)
    by (intro sum.cong refl) (use multipl-div-mono[of f - a] assms(4,6-8)  $\exists$  in
force)
  also have ... =  $\operatorname{moebius-mu} N * h N * f a * (\sum d \mid d \operatorname{dvd} a \wedge \operatorname{coprime} N d.$ 
moebius-mu d * (h d div f d))
    by (simp add: sum-distrib-left algebra-simps)
  finally show ?thesis by blast
  qed
  also have ... =
     $\operatorname{moebius-mu} N * h N * f a * (\prod p \in \{p. p \in \operatorname{prime-factors} a \wedge \neg (p \operatorname{dvd}$ 
N}). 1 - (h p div f p))
  proof -
    have multiplicative-function  $(\lambda d. h d \operatorname{div} f d)$ 
      using multiplicative-function-divide
      comp-to-mult
      assms(4,5) by blast
    then have  $(\sum d \mid d \operatorname{dvd} a \wedge \operatorname{coprime} N d. \operatorname{moebius-mu} d * (h d \operatorname{div} f d)) =$ 
 $(\prod p \in \{p. p \in \operatorname{prime-factors} a \wedge \neg (p \operatorname{dvd} N)\}. 1 - (h p \operatorname{div} f p))$ 
      using sum-divisors-moebius-mu-times-multiplicative-revisited[
        of  $(\lambda d. h d \operatorname{div} f d) a N$ ]
        assms(8) Ngr0 3 by blast
    then show ?thesis by argo
  qed
  also have ... =  $f(a) * \operatorname{moebius-mu}(N) * h(N) *$ 
 $((\prod p \in \{p. p \in \operatorname{prime-factors} (a*N)\}. 1 - (h p \operatorname{div} f p)) \operatorname{div}$ 
 $(\prod p \in \{p. p \in \operatorname{prime-factors} N\}. 1 - (h p \operatorname{div} f p)))$ 
  proof -
    have  $\{p. p \in \operatorname{prime-factors} a \wedge \neg p \operatorname{dvd} N\} =$ 
 $(\{p. p \in \operatorname{prime-factors} (a*N)\} - \{p. p \in \operatorname{prime-factors} N\})$ 
      using p-div-set[of a N] by blast
    then have eq2:  $(\prod p \in \{p. p \in \operatorname{prime-factors} a \wedge \neg p \operatorname{dvd} N\}. 1 - h p / f p) =$ 
 $\operatorname{prod} (\lambda p. 1 - h p / f p) (\{p. p \in \operatorname{prime-factors} (a*N)\} - \{p. p \in \operatorname{prime-factors}$ 
N})
      by auto
    also have eq: ... =  $\operatorname{prod} (\lambda p. 1 - h p / f p) \{p. p \in \operatorname{prime-factors} (a*N)\} \operatorname{div}$ 
 $\operatorname{prod} (\lambda p. 1 - h p / f p) \{p. p \in \operatorname{prime-factors} N\}$ 
  proof (intro prod-div-sub, simp, simp, simp add: 3 Ngr0 dvd-prime-factors, simp, standard)
    fix b
    assume b  $\in \#$  prime-factorization N
    then have p-b: prime b using in-prime-factors-iff by blast
    then show  $f b = 0 \vee h b \neq f b$  using assms(6)[OF p-b] by auto
  qed

```


also have $\dots = (\prod_{p \in \{p. p \in \text{prime-factors } (a*N)\}}. 1 - (h\ p\ \text{div}\ f\ p))\ \text{div}$
 $(\prod_{p \in \{p. p \in \text{prime-factors } N\}}. 1 - (h\ p\ \text{div}\ f\ p))$ **by** *blast*
finally have $(\prod_{p \in \{p. p \in \text{prime-factors } a \wedge \neg p\ \text{dvd}\ N\}}. 1 - h\ p / f\ p) =$
 $(\prod_{p \in \{p. p \in \text{prime-factors } (a*N)\}}. 1 - (h\ p\ \text{div}\ f\ p))\ \text{div}$
 $(\prod_{p \in \{p. p \in \text{prime-factors } N\}}. 1 - (h\ p\ \text{div}\ f\ p))$
using *eq eq2* **by** *auto*
then show *?thesis* **by** *simp*
qed
also have $\dots = f(a) * \text{moebius-mu}(N) * h(N) * (F(k)\ \text{div}\ f(k)) * (f(N)\ \text{div}$
 $F(N))$
(is *?a = ?b*
proof –
have $F(N) = (f\ N) * (\prod_{p \in \text{prime-factors } N}. 1 - (h\ p\ \text{div}\ f\ p))$
unfolding *F-def g-def*
by (*intro dirichlet-prod-completely-multiplicative-left*) (*auto simp add: Ngr0*
assms(4-6))
then have *eq-1*: $(\prod_{p \in \text{prime-factors } N}. 1 - (h\ p\ \text{div}\ f\ p)) =$
 $F\ N\ \text{div}\ f\ N$ **using** *2 f-N-not-z* **by** *simp*
have $F(k) = (f\ k) * (\prod_{p \in \text{prime-factors } k}. 1 - (h\ p\ \text{div}\ f\ p))$
unfolding *F-def g-def*
by (*intro dirichlet-prod-completely-multiplicative-left*) (*auto simp add: assms(4-7)*)
then have *eq-2*: $(\prod_{p \in \text{prime-factors } k}. 1 - (h\ p\ \text{div}\ f\ p)) =$
 $F\ k\ \text{div}\ f\ k$ **using** *2 f-k-not-z* **by** *simp*

have *?a = f a * moebius-mu N * h N **
 $(\prod_{p \in \text{prime-factors } k}. 1 - (h\ p\ \text{div}\ f\ p))\ \text{div}$
 $(\prod_{p \in \text{prime-factors } N}. 1 - (h\ p\ \text{div}\ f\ p))$
using *2* **by** (*simp add: algebra-simps*)
also have $\dots = f\ a * \text{moebius-mu}\ N * h\ N * ((F\ k\ \text{div}\ f\ k)\ \text{div}\ (F\ N\ \text{div}\ f\ N))$
by (*simp add: eq-1 eq-2*)
finally show *?thesis* **by** *simp*
qed
also have $\dots = \text{moebius-mu}\ N * h\ N * ((F\ k * f\ a * f\ N)\ \text{div}\ (F\ N * f\ k))$
by (*simp add: algebra-simps*)
also have $\dots = \text{moebius-mu}\ N * h\ N * ((F\ k * f(a*N))\ \text{div}\ (F\ N * f\ k))$
proof –
have $f\ a * f\ N = f\ (a*N)$
proof (*cases a = 1 ∨ N = 1*)
case *True*
then show *?thesis*
using *assms(4) completely-multiplicative-function-def[of f]*
by *auto*
next
case *False*
then show *?thesis*
using *2 assms(4) completely-multiplicative-function-def[of f]*
 $Ngr0\ 3$ **by** *auto*
qed
then show *?thesis* **by** *simp*

qed
also have $\dots = \text{moebius-mu } N * h \ N * ((F \ k * f(k)) \ \text{div} \ (F \ N * f \ k))$
using \mathcal{Q} **by** *blast*
also have $\dots = g(N) * (F \ k \ \text{div} \ F \ N)$
using *f-k-not-z g-def* **by** *simp*
also have $\dots = (F(k)*g(N)) \ \text{div} \ (F(N))$ **by** *auto*
finally show *?thesis* **by** *simp*
qed

lemma *totient-conv-moebius-mu-of-nat*:
 $\text{of-nat} \ (\text{totient } n) = \text{dirichlet-prod} \ \text{moebius-mu} \ \text{of-nat } n$
proof (*cases* $n = 0$)
case *False*
show *?thesis*
by (*rule moebius-inversion*)
(insert False, simp-all add: of-nat-sum [symmetric] totient-divisor-sum del:
of-nat-sum)
qed *simp-all*

corollary *ramanujan-sum-k-n-dirichlet-expr*:

fixes $k \ n :: \text{nat}$
assumes $k > 0 \ n > 0$
shows $c \ k \ n = \text{of-nat} \ (\text{totient } k) * \text{moebius-mu} \ (k \ \text{div} \ \text{gcd } n \ k) \ \text{div} \ \text{of-nat} \ (\text{totient} \ (k \ \text{div} \ \text{gcd } n \ k))$

proof –

define $f :: \text{nat} \Rightarrow \text{complex}$
where $f \equiv \text{of-nat}$
define $F :: \text{nat} \Rightarrow \text{complex}$
where $F \equiv (\lambda d. \ \text{dirichlet-prod} \ f \ \text{moebius-mu} \ d)$
define $g :: \text{nat} \Rightarrow \text{complex}$
where $g \equiv (\lambda l. \ \text{moebius-mu} \ l)$
define N **where** $N \equiv k \ \text{div} \ \text{gcd } n \ k$
define $h :: \text{nat} \Rightarrow \text{complex}$
where $h \equiv (\lambda x. \ (\text{if } x = 0 \ \text{then } 0 \ \text{else } 1))$

have *F-is-totient-k*: $F \ k = \text{totient } k$
by (*simp add: F-def f-def dirichlet-prod-commutes totient-conv-moebius-mu-of-nat[of k]*)
have *F-is-totient-N*: $F \ N = \text{totient } N$
by (*simp add: F-def f-def dirichlet-prod-commutes totient-conv-moebius-mu-of-nat[of N]*)

have $c \ k \ n = s \ \text{id} \ \text{moebius-mu} \ k \ n$
using *ramanujan-sum-conv-gen-ramanujan-sum assms* **by** *blast*
also have $\dots = s \ f \ g \ k \ n$
unfolding *f-def g-def* **by** *auto*
also have $g = (\lambda k. \ \text{moebius-mu} \ k * h \ k)$

```

  by (simp add: fun-eq-iff h-def g-def)
also have multiplicative-function h
  unfolding h-def by standard auto
hence s f ( $\lambda k. \text{moebius-mu } k * h k$ ) k n =
  dirichlet-prod of-nat ( $\lambda k. \text{moebius-mu } k * h k$ ) k *
  (moebius-mu (k div gcd n k) * h (k div gcd n k)) /
  dirichlet-prod of-nat ( $\lambda k. \text{moebius-mu } k * h k$ ) (k div gcd n k)
  unfolding f-def using assms mult-of-nat-c
  by (intro gen-ramanujan-sum-dirichlet-expr) (auto simp: h-def)
also have ... = of-nat (totient k) * moebius-mu (k div gcd n k) / of-nat (totient
(k div gcd n k))
  using F-is-totient-k F-is-totient-N by (auto simp: h-def F-def N-def f-def)
  finally show ?thesis .
qed

```

```

no-notation ramanujan-sum (c)
no-notation gen-ramanujan-sum (s)

```

end

```

theory Gauss-Sums
imports
  HOL-Algebra.Coset
  HOL-Real-Asymp.Real-Asymp
  Ramanujan-Sums
begin

```

7 Gauss sums

```

bundle vec-lambda-notation
begin
notation vec-lambda (binder  $\chi$  10)
end

bundle no-vec-lambda-notation
begin
no-notation vec-lambda (binder  $\chi$  10)
end

unbundle no-vec-lambda-notation

```

7.1 Definition and basic properties

```

context dcharacter
begin

```

```

lemma dir-periodic-arithmetic: periodic-arithmetic  $\chi$  n
  unfolding periodic-arithmetic-def by (simp add: periodic)

```

definition *gauss-sum* $k = (\sum m = 1..n . \chi(m) * \text{unity-root } n (m*k))$

lemma *gauss-sum-periodic*:

periodic-arithmetic $(\lambda n. \text{gauss-sum } n) n$

proof –

have *periodic-arithmetic* χn **using** *dir-periodic-arithmetic* **by** *simp*

let $?h = \lambda m k. \chi(m) * \text{unity-root } n (m*k)$

{fix $m :: \text{nat}$

have *periodic-arithmetic* $(\lambda k. \text{unity-root } n (m*k)) n$

using *unity-periodic-arithmetic-mult*[*of n m*] **by** *simp*

have *periodic-arithmetic* $(?h m) n$

using *scalar-mult-periodic-arithmetic*[*OF* $(\text{periodic-arithmetic } (\lambda k. \text{unity-root } n (m*k)) n)$]

by *blast*}

then have *per-all*: $\forall m \in \{1..n\}. \text{periodic-arithmetic } (?h m) n$ **by** *blast*

have *periodic-arithmetic* $(\lambda k. (\sum m = 1..n . \chi(m) * \text{unity-root } n (m*k))) n$

using *fin-sum-periodic-arithmetic-set*[*OF per-all*] **by** *blast*

then show *?thesis*

unfolding *gauss-sum-def* **by** *blast*

qed

lemma *ramanujan-sum-conv-gauss-sum*:

assumes $\chi = \text{principal-dchar } n$

shows *ramanujan-sum* $n k = \text{gauss-sum } k$

proof –

{fix m

from *assms*

have *1*: *coprime* $m n \implies \chi(m) = 1$ **and**

2: $\neg \text{coprime } m n \implies \chi(m) = 0$

unfolding *principal-dchar-def* **by** *auto*}

note *eq = this*

have *gauss-sum* $k = (\sum m = 1..n . \chi(m) * \text{unity-root } n (m*k))$

unfolding *gauss-sum-def* **by** *simp*

also have $\dots = (\sum m \mid m \in \{1..n\} \wedge \text{coprime } m n . \chi(m) * \text{unity-root } n (m*k))$

by *(rule sum.mono-neutral-right, simp, blast, simp add: eq)*

also have $\dots = (\sum m \mid m \in \{1..n\} \wedge \text{coprime } m n . \text{unity-root } n (m*k))$

by *(simp add: eq)*

also have $\dots = \text{ramanujan-sum } n k$ **unfolding** *ramanujan-sum-def* **by** *blast*

finally show *?thesis ..*

qed

lemma *cnj-mult-self*:

assumes *coprime* $k n$

shows *cnj* $(\chi k) * \chi k = 1$

proof –

have *cnj* $(\chi k) * \chi k = \text{norm } (\chi k) ^2$

by *(simp add: mult.commute complex-mult-cnj cmod-def)*

also have $\dots = 1$
using *norm[of k] assms by simp*
finally show *?thesis* .
qed

Theorem 8.9

theorem *gauss-sum-reduction:*

assumes *coprime k n*

shows *gauss-sum k = cnj (χ k) * gauss-sum 1*

proof –

from *n* **have** *n-pos: n > 0* **by** *simp*

have *gauss-sum k = (∑ r = 1..n . χ(r) * unity-root n (r*k))*

unfolding *gauss-sum-def* **by** *simp*

also have $\dots = (\sum r = 1..n . cnj (\chi(k)) * \chi k * \chi r * unity-root n (r*k))$

using *assms by (intro sum.cong) (auto simp: cnj-mult-self)*

also have $\dots = (\sum r = 1..n . cnj (\chi(k)) * \chi (k*r) * unity-root n (r*k))$

by *(intro sum.cong) auto*

also have $\dots = cnj (\chi(k)) * (\sum r = 1..n . \chi (k*r) * unity-root n (r*k))$

by *(simp add: sum-distrib-left algebra-simps)*

also have $\dots = cnj (\chi(k)) * (\sum r = 1..n . \chi r * unity-root n r)$

proof –

have *1: periodic-arithmetic (λr. χ r * unity-root n r) n*

using *dir-periodic-arithmetic unity-periodic-arithmetic mult-periodic-arithmetic*

by *blast*

have $(\sum r = 1..n . \chi (k*r) * unity-root n (r*k)) =$

$(\sum r = 1..n . \chi (r) * unity-root n r)$

using *periodic-arithmetic-remove-homothecy[OF assms(1) 1 n-pos]*

by *(simp add: algebra-simps n)*

then show *?thesis* **by** *argo*

qed

also have $\dots = cnj (\chi(k)) * gauss-sum 1$

using *gauss-sum-def* **by** *simp*

finally show *?thesis* .

qed

The following variant takes an integer argument instead.

definition *gauss-sum-int k = (∑ m=1..n. χ m * unity-root n (int m*k))*

sublocale *gauss-sum-int: periodic-fun-simple gauss-sum-int int n*

proof

fix *k*

show *gauss-sum-int (k + int n) = gauss-sum-int k*

by *(simp add: gauss-sum-int-def ring-distrib unity-root-add)*

qed

lemma *gauss-sum-int-cong:*

assumes $[a = b] \pmod{int n}$

shows *gauss-sum-int a = gauss-sum-int b*

proof –

from *assms* **obtain** *k* **where** $k: b = a + \text{int } n * k$
by (*subst (asm) cong-iff-lin*) *auto*
thus *?thesis*
using *gauss-sum-int.plus-of-int[of a k]* **by** (*auto simp: algebra-simps*)
qed

lemma *gauss-sum-conv-gauss-sum-int:*
 $\text{gauss-sum } k = \text{gauss-sum-int } (\text{int } k)$
unfolding *gauss-sum-def gauss-sum-int-def* **by** *auto*

lemma *gauss-sum-int-conv-gauss-sum:*
 $\text{gauss-sum-int } k = \text{gauss-sum } (\text{nat } (k \text{ mod } n))$
proof –
have $\text{gauss-sum } (\text{nat } (k \text{ mod } n)) = \text{gauss-sum-int } (\text{int } (\text{nat } (k \text{ mod } n)))$
by (*simp add: gauss-sum-conv-gauss-sum-int*)
also have $\dots = \text{gauss-sum-int } k$
using *n*
by (*intro gauss-sum-int-cong*) (*auto simp: cong-def*)
finally show *?thesis ..*
qed

lemma *gauss-int-periodic: periodic-arithmetic gauss-sum-int n*
unfolding *periodic-arithmetic-def gauss-sum-int-conv-gauss-sum* **by** *simp*

proposition *dcharacter-fourier-expansion:*
 $\chi \ m = (\sum k=1..n. 1 / n * \text{gauss-sum-int } (-k) * \text{unity-root } n \ (m*k))$
proof –
define *g* **where** $g = (\lambda x. 1 / \text{of-nat } n * (\sum m < n. \chi \ m * \text{unity-root } n \ (- \text{int } x * \text{int } m)))$
have *per: periodic-arithmetic $\chi \ n$* **using** *dir-periodic-arithmetic* **by** *simp*
have $\chi \ m = (\sum k < n. g \ k * \text{unity-root } n \ (m * \text{int } k))$
using *fourier-expansion-periodic-arithmetic(2)[OF - per, of m] n* **by** (*auto simp: g-def*)
also have $\dots = (\sum k = 1..n. g \ k * \text{unity-root } n \ (m * \text{int } k))$
proof –
have *g-per: periodic-arithmetic g n*
using *fourier-expansion-periodic-arithmetic(1)[OF - per] n* **by** (*simp add: g-def*)
have *fact-per: periodic-arithmetic $(\lambda k. g \ k * \text{unity-root } n \ (\text{int } m * \text{int } k)) \ n$*
using *mult-periodic-arithmetic[OF g-per] unity-periodic-arithmetic-mult* **by** *auto*
show *?thesis*
proof –
have $(\sum k < n. g \ k * \text{unity-root } n \ (\text{int } m * \text{int } k)) = (\sum l = 0..n - \text{Suc } 0. g \ l * \text{unity-root } n \ (\text{int } m * \text{int } l))$
using *n* **by** (*intro sum.cong*) *auto*
also have $\dots = (\sum l = \text{Suc } 0..n. g \ l * \text{unity-root } n \ (\text{int } m * \text{int } l))$
using *periodic-arithmetic-sum-periodic-arithmetic-shift[OF fact-per, of 1] n*
by *auto*

```

    finally show ?thesis by simp
  qed
qed
also have ... = (∑ k = 1..n. (1 / of-nat n) * gauss-sum-int (-k) * unity-root
n (m*k))
  proof -
    {fix k :: nat
  have shift: (∑ m < n. χ m * unity-root n (- int k * int m)) =
    (∑ m = 1..n. χ m * unity-root n (- int k * int m))
  proof -
    have per-unit: periodic-arithmetic (λm. unity-root n (- int k * int m)) n
      using unity-periodic-arithmetic-mult by blast
    then have prod-per: periodic-arithmetic (λm. χ m * unity-root n (- int k *
int m)) n
      using per mult-periodic-arithmetic by blast
    show ?thesis
  proof -
    have (∑ m < n. χ m * unity-root n (- int k * int m)) =
      (∑ l = 0..n - Suc 0. χ l * unity-root n (- int k * int l))
      using n by (intro sum.cong) auto
    also have ... = (∑ m = 1..n. χ m * unity-root n (- int k * int m))
      using periodic-arithmetic-sum-periodic-arithmetic-shift[OF prod-per, of 1]
n by auto
    finally show ?thesis by simp
  qed
qed
have g k = 1 / of-nat n *
  (∑ m < n. χ m * unity-root n (- int k * int m))
  using g-def by auto
also have ... = 1 / of-nat n *
  (∑ m = 1..n. χ m * unity-root n (- int k * int m))
  using shift by simp
also have ... = 1 / of-nat n * gauss-sum-int (-k)
  unfolding gauss-sum-int-def
  by (simp add: algebra-simps)
finally have g k = 1 / of-nat n * gauss-sum-int (-k) by simp}
note g-expr = this
  show ?thesis
    by (rule sum.cong, simp, simp add: g-expr)
  qed
  finally show ?thesis by auto
qed

```

7.2 Separability

definition *separable* $k \longleftrightarrow \text{gauss-sum } k = \text{cnj } (\chi \ k) * \text{gauss-sum } 1$

corollary *gauss-coprime-separable*:

assumes *coprime* $k \ n$

shows *separable k*
using *gauss-sum-reduction[OF assms]* **unfolding** *separable-def* **by** *simp*

Theorem 8.10

theorem *global-separability-condition:*

$(\forall n > 0. \text{separable } n) \longleftrightarrow (\forall k > 0. \neg \text{coprime } k \ n \longrightarrow \text{gauss-sum } k = 0)$

proof –

{**fix** *k*
assume $\neg \text{coprime } k \ n$
then have $\chi(k) = 0$ **by** (*simp add: eq-zero*)
then have $\text{cnj } (\chi \ k) = 0$ **by** *blast*
then have $\text{separable } k \longleftrightarrow \text{gauss-sum } k = 0$
unfolding *separable-def* **by** *auto*}
note *not-case = this*

show *?thesis*

using *gauss-coprime-separable not-case separable-def* **by** *blast*

qed

lemma *of-real-moebius-mu [simp]: of-real (moebius-mu k) = moebius-mu k*
by (*simp add: moebius-mu-def*)

corollary *principal-not-totally-separable:*

assumes $\chi = \text{principal-dchar } n$
shows $\neg(\forall k > 0. \text{separable } k)$

proof –

have *n-pos: n > 0* **using** *n* **by** *simp*
have *tot-0: totient n ≠ 0* **by** (*simp add: n-pos*)
have *moebius-mu (n div gcd n n) ≠ 0* **by** (*simp add: (n > 0)*)
then have *moeb-0: ∃ k. moebius-mu (n div gcd k n) ≠ 0* **by** *blast*

have *lem: gauss-sum k = totient n * moebius-mu (n div gcd k n) / totient (n div gcd k n)*

if $k > 0$ **for** *k*

proof –

have *gauss-sum k = ramanujan-sum n k*
using *ramanujan-sum-conv-gauss-sum[OF assms(1)] ..*

also have $\dots = \text{totient } n * \text{moebius-mu } (n \ \text{div} \ \text{gcd } k \ n) / (\text{totient } (n \ \text{div} \ \text{gcd } k \ n))$

by (*simp add: ramanujan-sum-k-n-dirichlet-expr[OF n-pos that]*)

finally show *?thesis .*

qed

have *2: ¬ coprime n n* **using** *n* **by** *auto*

have *3: gauss-sum n ≠ 0*

using *lem[OF n-pos] tot-0 moebius-mu-1* **by** *simp*

from *n-pos 2 3* **have**

$\exists k > 0. \neg \text{coprime } k \ n \wedge \text{gauss-sum } k \neq 0$ **by** *blast*

then obtain *k* **where** $k > 0 \wedge \neg \text{coprime } k \ n \wedge \text{gauss-sum } k \neq 0$ **by** *blast*

note *right-not-zero = this*

have $\text{cnj } (\chi \ k) * \text{gauss-sum } 1 = 0$ **if** $\neg \text{coprime } k \ n$ **for** k
using *that assms* **by** (*simp add: principal-dchar-def*)
then show *?thesis*
unfolding *separable-def* **using** *right-not-zero* **by** *auto*
qed

Theorem 8.11

theorem *gauss-sum-1-mod-square-eq-k*:
assumes $(\forall k. k > 0 \longrightarrow \text{separable } k)$
shows $\text{norm } (\text{gauss-sum } 1) ^ 2 = \text{real } n$
proof –
have $(\text{norm } (\text{gauss-sum } 1)) ^ 2 = \text{gauss-sum } 1 * \text{cnj } (\text{gauss-sum } 1)$
using *complex-norm-square* **by** *blast*
also have $\dots = \text{gauss-sum } 1 * (\sum m = 1..n. \text{cnj } (\chi(m)) * \text{unity-root } n \ (-m))$
proof –
have $\text{cnj } (\text{gauss-sum } 1) = (\sum m = 1..n. \text{cnj } (\chi(m)) * \text{unity-root } n \ (-m))$
unfolding *gauss-sum-def* **by** (*simp add: unity-root-uminus*)
then show *?thesis* **by** *argo*
qed
also have $\dots = (\sum m = 1..n. \text{gauss-sum } 1 * \text{cnj } (\chi(m)) * \text{unity-root } n \ (-m))$
by (*subst sum-distrib-left*)(*simp add: algebra-simps*)
also have $\dots = (\sum m = 1..n. \text{gauss-sum } m * \text{unity-root } n \ (-m))$
proof (*rule sum.cong,simp*)
fix x
assume $as: x \in \{1..n\}$
show $\text{gauss-sum } 1 * \text{cnj } (\chi \ x) * \text{unity-root } n \ (-x) =$
 $\text{gauss-sum } x * \text{unity-root } n \ (-x)$
using *assms(1)* **unfolding** *separable-def*
by (*rule allE[of - x]*) (*use as in auto*)
qed
also have $\dots = (\sum m = 1..n. (\sum r = 1..n. \chi \ r * \text{unity-root } n \ (r*m) * \text{unity-root } n \ (-m)))$
unfolding *gauss-sum-def*
by (*rule sum.cong,simp,rule sum-distrib-right*)
also have $\dots = (\sum m = 1..n. (\sum r = 1..n. \chi \ r * \text{unity-root } n \ (m*(r-1))))$
by (*intro sum.cong refl*) (*auto simp: unity-root-diff of-nat-diff unity-root-uminus field-simps*)
also have $\dots = (\sum r=1..n. (\sum m=1..n. \chi(r) * \text{unity-root } n \ (m*(r-1))))$
by (*rule sum.swap*)
also have $\dots = (\sum r=1..n. \chi(r) * (\sum m=1..n. \text{unity-root } n \ (m*(r-1))))$
by (*rule sum.cong, simp, simp add: sum-distrib-left*)
also have $\dots = (\sum r=1..n. \chi(r) * \text{unity-root-sum } n \ (r-1))$
proof (*intro sum.cong refl*)
fix x
assume $x \in \{1..n\}$
then have $1: \text{periodic-arithmetic } (\lambda m. \text{unity-root } n \ (\text{int } (m * (x - 1)))) \ n$
using *unity-periodic-arithmetic-mult[of n x-1]*
by (*simp add: mult commute*)

have $(\sum m = 1..n. \text{unity-root } n \text{ (int } (m * (x - 1)))) =$
 $(\sum m = 0..n-1. \text{unity-root } n \text{ (int } (m * (x - 1))))$
using *periodic-arithmetic-sum-periodic-arithmetic-shift*[*OF 1 -, of 1*] *n* **by**
simp
also have $\dots = \text{unity-root-sum } n \text{ (} x-1 \text{)}$
using *n unfolding unity-root-sum-def* **by** (*intro sum.cong*) (*auto simp:*
mult-ac)
finally have $(\sum m = 1..n. \text{unity-root } n \text{ (int } (m * (x - 1)))) =$
 $\text{unity-root-sum } n \text{ (int } (x - 1)) .$
then show $\chi x * (\sum m = 1..n. \text{unity-root } n \text{ (int } (m * (x - 1)))) =$
 $\chi x * \text{unity-root-sum } n \text{ (int } (x - 1))$ **by** *argo*
qed
also have $\dots = (\sum r \in \{1\}. \chi r * \text{unity-root-sum } n \text{ (int } (r - 1)))$
using *n unity-root-sum-nonzero-iff int-ops(6)*
by (*intro sum.mono-neutral-right*) *auto*
also have $\dots = \chi 1 * n$ **using** *n* **by** *simp*
also have $\dots = n$ **by** *simp*
finally show *?thesis*
using *of-real-eq-iff* **by** *fastforce*
qed

Theorem 8.12

theorem *gauss-sum-nonzero-noncoprime-necessary-condition:*

assumes *gauss-sum* $k \neq 0 \neg \text{coprime } k n k > 0$

defines $d \equiv n \text{ div } \text{gcd } k n$

assumes *coprime* $a n [a = 1] \text{ (mod } d)$

shows $d \text{ dvd } n d < n \chi a = 1$

proof –

show $d \text{ dvd } n$

unfolding *d-def* **using** *n* **by** (*subst div-dvd-iff-mult*) *auto*

from *assms(2)* **have** $\text{gcd } k n \neq 1$ **by** *blast*

then have $\text{gcd } k n > 1$ **using** *assms(3,4)* **by** (*simp add: nat-neq-iff*)

with *n* **show** $d < n$ **by** (*simp add: d-def*)

have *periodic-arithmetic* $(\lambda r. \chi (r) * \text{unity-root } n \text{ (} k*r \text{)}) n$

using *mult-periodic-arithmetic*[*OF dir-periodic-arithmetic unity-periodic-arithmetic-mult*]

by *auto*

then have *1: periodic-arithmetic* $(\lambda r. \chi (r) * \text{unity-root } n \text{ (} r*k \text{)}) n$

by (*simp add: algebra-simps*)

have *gauss-sum* $k = (\sum m = 1..n . \chi(m) * \text{unity-root } n \text{ (} m*k \text{)})$

unfolding *gauss-sum-def* **by** *blast*

also have $\dots = (\sum m = 1..n . \chi(m*a) * \text{unity-root } n \text{ (} m*a*k \text{)})$

using *periodic-arithmetic-remove-homothecy*[*OF assms(5) 1*] *n* **by** *auto*

also have $\dots = (\sum m = 1..n . \chi(m*a) * \text{unity-root } n \text{ (} m*k \text{)})$

proof (*intro sum.cong refl*)

fix *m*

from *assms(6)* **obtain** *b* **where** $a = 1 + b*d$

using $\langle d < n \rangle$ *assms(5)* *cong-to-1'-nat* **by** *auto*

then have $m*a*k = m*k + m*b*(n \text{ div } \text{gcd } k \ n)*k$
by (*simp add: algebra-simps d-def*)
also have $\dots = m*k + m*b*n*(k \text{ div } \text{gcd } k \ n)$
by (*simp add: div-mult-swap dvd-div-mult*)
also obtain p **where** $\dots = m*k + m*b*n*p$ **by** *blast*
finally have $m*a*k = m*k + m*b*p*n$ **by** *simp*
then have $1: m*a*k \text{ mod } n = m*k \text{ mod } n$
using *mod-mult-self1* **by** *simp*
then have $\text{unity-root } n \ (m * a * k) = \text{unity-root } n \ (m * k)$
proof –
have $\text{unity-root } n \ (m * a * k) = \text{unity-root } n \ ((m * a * k) \text{ mod } n)$
using *unity-root-mod[of n] zmod-int* **by** *simp*
also have $\dots = \text{unity-root } n \ (m * k)$
using *unity-root-mod[of n] zmod-int 1* **by** *presburger*
finally show *?thesis* **by** *blast*
qed
then show $\chi \ (m * a) * \text{unity-root } n \ (\text{int } (m * a * k)) =$
 $\chi \ (m * a) * \text{unity-root } n \ (\text{int } (m * k))$ **by** *auto*
qed
also have $\dots = (\sum m = 1..n . \chi(a) * (\chi(m) * \text{unity-root } n \ (m*k)))$
by (*rule sum.cong, simp, subst mult, simp*)
also have $\dots = \chi(a) * (\sum m = 1..n . \chi(m) * \text{unity-root } n \ (m*k))$
by (*simp add: sum-distrib-left[symmetric]*)
also have $\dots = \chi(a) * \text{gauss-sum } k$
unfolding *gauss-sum-def* **by** *blast*
finally have $\text{gauss-sum } k = \chi(a) * \text{gauss-sum } k$ **by** *blast*
then show $\chi \ a = 1$
using *assms(1)* **by** *simp*
qed

7.3 Induced moduli and primitive characters

definition *induced-modulus* $d \longleftrightarrow d \text{ dvd } n \wedge (\forall a. \text{coprime } a \ n \wedge [a = 1] \ (\text{mod } d) \longrightarrow \chi \ a = 1)$

lemma *induced-modulus-dvd*: $\text{induced-modulus } d \Longrightarrow d \text{ dvd } n$
unfolding *induced-modulus-def* **by** *blast*

lemma *induced-modulusI* [*intro?*]:
 $d \text{ dvd } n \Longrightarrow (\bigwedge a. \text{coprime } a \ n \Longrightarrow [a = 1] \ (\text{mod } d) \Longrightarrow \chi \ a = 1) \Longrightarrow \text{induced-modulus } d$
unfolding *induced-modulus-def* **by** *auto*

lemma *induced-modulusD*: $\text{induced-modulus } d \Longrightarrow \text{coprime } a \ n \Longrightarrow [a = 1] \ (\text{mod } d) \Longrightarrow \chi \ a = 1$
unfolding *induced-modulus-def* **by** *blast*

lemma *zero-not-ind-mod*: $\neg \text{induced-modulus } 0$
unfolding *induced-modulus-def* **using** n **by** *simp*

lemma *div-gcd-dvd1*: $(a :: 'a :: \text{semiring-gcd}) \text{ div gcd } a \ b \ \text{dvd } a$
by $(\text{metis } \text{dvd-def } \text{dvd-div-mult-self } \text{gcd-dvd1})$

lemma *div-gcd-dvd2*: $(b :: 'a :: \text{semiring-gcd}) \text{ div gcd } a \ b \ \text{dvd } b$
by $(\text{metis } \text{div-gcd-dvd1 } \text{gcd.commute})$

lemma *g-non-zero-ind-mod*:

assumes $\text{gauss-sum } k \neq 0 \ \neg \text{coprime } k \ n \ k > 0$

shows $\text{induced-modulus } (n \ \text{div } \text{gcd } k \ n)$

proof

show $n \ \text{div } \text{gcd } k \ n \ \text{dvd } n$

by $(\text{metis } \text{dvd-div-mult-self } \text{dvd-triv-left } \text{gcd.commute } \text{gcd-dvd1})$

fix $a :: \text{nat}$

assume $\text{coprime } a \ n \ [a = 1] \ (\text{mod } n \ \text{div } \text{gcd } k \ n)$

thus $\chi \ a = 1$

using $\text{assms } n \ \text{gauss-sum-nonzero-noncoprime-necessary-condition}(3)$ **by** *auto*

qed

lemma *induced-modulus-modulus*: $\text{induced-modulus } n$

unfolding *induced-modulus-def*

proof $(\text{rule } \text{conjI}, \text{simp}, \text{safe})$

fix a

assume $[a = 1] \ (\text{mod } n)$

then have $a \ \text{mod } n = 1 \ \text{mod } n$

using *cong-def*[*of a 1 n*] **by** *blast*

also have $\dots = 1$

using *eq-zero-iff zero-eq-0* **by** *fastforce*

finally have $1: a \ \text{mod } n = 1$ **by** *simp*

have $\chi \ a = \chi \ (a \ \text{mod } n)$ **by** *simp*

also have $\dots = \chi \ 1$ **using** *cong-def 1* **by** *auto*

also have $\dots = 1$ **by** *simp*

finally show $\chi \ a = 1$ **by** *blast*

qed

Theorem 8.13

theorem *one-induced-iff-principal*:

$\text{induced-modulus } 1 \iff \chi = \text{principal-dchar } n$

proof

assume *induced-modulus 1*

then have $(\forall a. \text{coprime } a \ n \implies \chi \ a = 1)$

unfolding *induced-modulus-def* **by** *simp*

then show $\chi = \text{principal-dchar } n$

unfolding *principal-dchar-def* **using** *eq-zero* **by** *auto*

next

assume $as: \chi = \text{principal-dchar } n$

{**fix** a

assume $\text{coprime } a \ n$

then have $\chi a = 1$
using *principal-dchar-def as by simp*
then show *induced-modulus 1*
unfolding *induced-modulus-def by auto*
qed
end

locale *primitive-dchar* = *dcharacter* +
assumes *no-induced-modulus*: $\neg(\exists d < n. \textit{induced-modulus } d)$

locale *nonprimitive-dchar* = *dcharacter* +
assumes *induced-modulus*: $\exists d < n. \textit{induced-modulus } d$

lemma (**in** *nonprimitive-dchar*) *nonprimitive*: $\neg\textit{primitive-dchar } n \chi$
proof
assume *primitive-dchar* $n \chi$
then interpret *A*: *primitive-dchar* n *residue-mult-group* $n \chi$
by *auto*
from *A.no-induced-modulus induced-modulus* **show** *False* **by** *contradiction*
qed

lemma (**in** *dcharacter*) *primitive-dchar-iff*:
primitive-dchar $n \chi \longleftrightarrow \neg(\exists d < n. \textit{induced-modulus } d)$
unfolding *primitive-dchar-def primitive-dchar-axioms-def*
using *dcharacter-axioms* **by** *metis*

lemma (**in** *residues-nat*) *principal-not-primitive*:
 $\neg\textit{primitive-dchar } n (\textit{principal-dchar } n)$
unfolding *principal.primitive-dchar-iff*
using *principal.one-induced-iff-principal* n **by** *auto*

lemma (**in** *dcharacter*) *not-primitive-imp-nonprimitive*:
assumes $\neg\textit{primitive-dchar } n \chi$
shows *nonprimitive-dchar* $n \chi$
using *assms dcharacter-axioms*
unfolding *nonprimitive-dchar-def primitive-dchar-def*
primitive-dchar-axioms-def nonprimitive-dchar-axioms-def **by** *auto*

Theorem 8.14

theorem (**in** *dcharacter*) *prime-nonprincipal-is-primitive*:
assumes *prime* n
assumes $\chi \neq \textit{principal-dchar } n$
shows *primitive-dchar* $n \chi$
proof –
{fix m
assume *induced-modulus* m
then have $m = n$

using *assms prime-nat-iff induced-modulus-def*
one-induced-iff-principal by blast
then show *?thesis using primitive-dchar-iff by blast*
qed

Theorem 8.15

corollary (in *primitive-dchar*) *primitive-encoding*:

$\forall k > 0. \neg \text{coprime } k \ n \longrightarrow \text{gauss-sum } k = 0$

$\forall k > 0. \text{separable } k$

$\text{norm } (\text{gauss-sum } 1) \wedge 2 = n$

proof *safe*

show *1: gauss-sum k = 0 if k > 0 and $\neg \text{coprime } k \ n$ for k*

proof (*rule ccontr*)

assume *gauss-sum k \neq 0*

hence *induced-modulus (n div gcd k n)*

using *that by (intro g-non-zero-ind-mod) auto*

moreover have *n div gcd k n < n*

using *n that*

by (*meson coprime-iff-gcd-eq-1 div-eq-dividend-iff le-less-trans*

linorder-neqE-nat nat-dvd-not-less principal.div-gcd-dvd2 zero-le-one)

ultimately show *False using no-induced-modulus by blast*

qed

have ($\forall n > 0. \text{separable } n$)

unfolding *global-separability-condition by (auto intro!: 1)*

thus *separable n if n > 0 for n*

using *that by blast*

thus $\text{norm } (\text{gauss-sum } 1) \wedge 2 = n$

using *gauss-sum-1-mod-square-eq-k by blast*

qed

Theorem 8.16

lemma (in *dcharacter*) *induced-modulus-altdef1*:

induced-modulus d \longleftrightarrow

$d \text{ dvd } n \wedge (\forall a \ b. \text{coprime } a \ n \wedge \text{coprime } b \ n \wedge [a = b] \ (\text{mod } d) \longrightarrow \chi \ a = \chi \ b)$

proof

assume *1: induced-modulus d*

with *n have d: d dvd n d > 0*

by (*auto simp: induced-modulus-def intro: Nat.gr0I*)

show $d \text{ dvd } n \wedge (\forall a \ b. \text{coprime } a \ n \wedge \text{coprime } b \ n \wedge [a = b] \ (\text{mod } d) \longrightarrow \chi(a) = \chi(b))$

proof *safe*

fix *a b*

assume *2: coprime a n coprime b n [a = b] (mod d)*

show $\chi(a) = \chi(b)$

proof $-$

from *2(1) obtain a' where eq: [a*a' = 1] (mod n)*

using *cong-solve by blast*

from *this d have [a*a' = 1] (mod d)*

```

    using cong-dvd-modulus-nat by blast
  from this 1 have  $\chi(a*a') = 1$ 
    unfolding induced-modulus-def
    by (meson 2(2) eq cong-imp-coprime cong-sym coprime-divisors gcd-nat.refl
one-dvd)
  then have 3:  $\chi(a)*\chi(a') = 1$ 
    by simp

  from 2(3) have  $[a*a' = b*a'] \pmod d$ 
    by (simp add: cong-scalar-right)
  moreover have 4:  $[b*a' = 1] \pmod d$ 
    using  $\langle [a * a' = 1] \pmod d \rangle$  calculation cong-sym cong-trans by blast
  have  $\chi(b*a') = 1$ 
  proof -
    have coprime (b*a') n
      using 2(2) cong-imp-coprime[OF cong-sym[OF eq]] by simp
    then show ?thesis using 4 induced-modulus-def 1 by blast
  qed
  then have 4:  $\chi(b)*\chi(a') = 1$ 
    by simp
  from 3 4 show  $\chi(a) = \chi(b)$ 
    using mult-cancel-left
    by (cases  $\chi(a') = 0$ ) (fastforce simp add: field-simps)+
  qed
qed fact+
next
  assume *:  $d \text{ dvd } n \wedge (\forall a b. \text{coprime } a \ n \wedge \text{coprime } b \ n \wedge [a = b] \pmod d) \longrightarrow \chi a = \chi b$ 
  then have  $\forall a. \text{coprime } a \ n \wedge \text{coprime } 1 \ n \wedge [a = 1] \pmod d \longrightarrow \chi a = \chi 1$ 
    by blast
  then have  $\forall a. \text{coprime } a \ n \wedge [a = 1] \pmod d \longrightarrow \chi a = 1$ 
    using coprime-1-left by simp
  then show induced-modulus d
    unfolding induced-modulus-def using * by blast
  qed

```

Exercise 8.4

lemma *induced-modulus-altdef2-lemma:*

```

  fixes  $n \ a \ d \ q :: \text{nat}$ 
  defines  $q \equiv (\prod p \mid \text{prime } p \wedge p \text{ dvd } n \wedge \neg (p \text{ dvd } a). \ p)$ 
  defines  $m \equiv a + q * d$ 
  assumes  $n > 0$  coprime a d
  shows  $[m = a] \pmod d$  and coprime m n
proof (simp add: assms(2) cong-add-lcancel-0-nat cong-mult-self-right)
  have fin: finite {p. prime p  $\wedge$  p dvd n  $\wedge$   $\neg$  (p dvd a)} by (simp add: assms)
  { fix p
    assume 4: prime p p dvd m p dvd n
    have  $p = 1$ 
    proof (cases p dvd a)

```

```

case True
from this assms 4(2) have p dvd q*d
  by (simp add: dvd-add-right-iff)
then have a1: p dvd q  $\vee$  p dvd d
  using 4(1) prime-dvd-mult-iff by blast

have a2:  $\neg$  (p dvd q)
proof (rule ccontr, simp)
  assume p dvd q
  then have p dvd ( $\prod$  p | prime p  $\wedge$  p dvd n  $\wedge$   $\neg$  (p dvd a). p)
    unfolding assms by simp
  then have  $\exists x \in \{p. \text{prime } p \wedge p \text{ dvd } n \wedge \neg p \text{ dvd } a\}. p \text{ dvd } x$ 
    using prime-dvd-prod-iff[OF fin 4(1)] by simp
  then obtain x where c: p dvd x  $\wedge$  prime x  $\wedge$   $\neg$  x dvd a by blast
  then have p = x using 4(1) by (simp add: primes-dvd-imp-eq)
  then show False using True c by auto
qed
have a3:  $\neg$  (p dvd d)
  using True assms 4(1) coprime-def not-prime-unit by auto

from a1 a2 a3 show ?thesis by simp
next
case False
then have p dvd q
proof -
  have in-s:  $p \in \{p. \text{prime } p \wedge p \text{ dvd } n \wedge \neg p \text{ dvd } a\}$ 
    using False 4(3) 4(1) by simp
  show p dvd q
    unfolding assms using dvd-prodI[OF fin in-s ] by fast
qed
then have p dvd q*d by simp
then have p dvd a using 4(2) assms
  by (simp add: dvd-add-left-iff)
then show ?thesis using False by auto
qed
}
note lem = this
show coprime m n
proof (subst coprime-iff-gcd-eq-1)
  {fix a
  assume a dvd m a dvd n a  $\neq$  1
  {fix p
  assume prime p p dvd a
  then have p dvd m p dvd n
    using  $\langle a \text{ dvd } m \rangle \langle a \text{ dvd } n \rangle$  by auto
  from lem have p = a
    using not-prime-1  $\langle \text{prime } p \rangle \langle p \text{ dvd } m \rangle \langle p \text{ dvd } n \rangle$  by blast}
  then have prime a
    using prime-prime-factor[of a]  $\langle a \neq 1 \rangle$  by blast

```



```

    then have  $a = 1$  using  $lem \langle a \text{ dvd } m \rangle \langle a \text{ dvd } n \rangle$  by blast
    then have  $False$  using  $\langle a = 1 \rangle \langle a \neq 1 \rangle$  by blast
  }
  then show  $gcd \ m \ n = 1$  by blast
qed
qed

```

Theorem 8.17

The case $d = 1$ is exactly the case described in $dcharacter \ ?n \ ?\chi \implies dcharacter.induced-modulus \ ?n \ ?\chi \ 1 = (\ ?\chi = principal-dchar \ ?n)$.

theorem (in $dcharacter$) $induced-modulus-altdef2$:

assumes $d \text{ dvd } n \ d \neq 1$

defines $\chi_1 \equiv principal-dchar \ n$

shows $induced-modulus \ d \longleftrightarrow (\exists \Phi. dcharacter \ d \ \Phi \wedge (\forall k. \chi \ k = \Phi \ k * \chi_1 \ k))$

proof

from n **have** $n\text{-pos}: n > 0$ **by** $simp$

assume $as\text{-im}: induced-modulus \ d$

define f **where**

```

 $f \equiv (\lambda k. k +$ 
  (if  $k = 1$  then
    0
    else  $(prod \ id \ \{p. \ prime \ p \wedge p \text{ dvd } n \wedge \neg (p \text{ dvd } k)\}) * d$ )
  )

```

have $[simp]: f \ (Suc \ 0) = 1$ **unfolding** $f\text{-def}$ **by** $simp$

{

fix k

assume $as: coprime \ k \ d$

hence $[f \ k = k] \ (mod \ d) \ coprime \ (f \ k) \ n$

using $induced-modulus-altdef2\text{-lemma}[OF \ n\text{-pos} \ as]$ **by** $(simp\text{-all} \ add: \ f\text{-def})$

}

note $m\text{-prop} = this$

define Φ **where**

$\Phi \equiv (\lambda n. (if \ \neg \ coprime \ n \ d \ then \ 0 \ else \ \chi(f \ n)))$

have $\Phi\text{-1}: \Phi \ 1 = 1$

unfolding $\Phi\text{-def}$ **by** $simp$

from $assms(1,2)$ n **have** $d > 0$ **by** $(intro \ Nat.gr0I) \ auto$

from $induced-modulus-altdef1 \ assms(1) \ \langle d > 0 \rangle \ as\text{-im}$

have $b: (\forall a \ b. \ coprime \ a \ n \wedge \ coprime \ b \ n \wedge$
 $[a = b] \ (mod \ d) \longrightarrow \chi \ a = \chi \ b)$ **by** $blast$

have $\Phi\text{-periodic}: \forall a. \Phi \ (a + d) = \Phi \ a$

proof

fix a

have $gcd \ (a+d) \ d = gcd \ a \ d$ **by** $auto$

then **have** $cop: coprime \ (a+d) \ d = coprime \ a \ d$

```

    using coprime-iff-gcd-eq-1 by presburger
  show  $\Phi (a + d) = \Phi a$ 
  proof (cases coprime a d)
    case True
      from True cop have cop-ad: coprime (a+d) d by blast
      have p1:  $[f (a+d) = f a] \pmod d$ 
        using m-prop(1)[of a+d, OF cop-ad]
          m-prop(1)[of a, OF True] by (simp add: cong-def)
      have p2: coprime (f (a+d)) n coprime (f a) n
        using m-prop(2)[of a+d, OF cop-ad]
          m-prop(2)[of a, OF True] by blast+
      from b p1 p2 have eq:  $\chi (f (a + d)) = \chi (f a)$  by blast
      show ?thesis
        unfolding  $\Phi$ -def
        by (subst cop,simp,safe, simp add: eq)
    next
      case False
      then show ?thesis unfolding  $\Phi$ -def by (subst cop,simp)
  qed
qed

have  $\Phi$ -mult:  $\forall a b. a \in \text{totatives } d \longrightarrow$ 
   $b \in \text{totatives } d \longrightarrow \Phi (a * b) = \Phi a * \Phi b$ 
proof (safe)
  fix a b
  assume a  $\in$  totatives d b  $\in$  totatives d
  consider (ab) coprime a d  $\wedge$  coprime b d |
    (a) coprime a d  $\wedge$   $\neg$  coprime b d |
    (b) coprime b d  $\wedge$   $\neg$  coprime a d |
    (n)  $\neg$  coprime a d  $\wedge$   $\neg$  coprime b d by blast
  then show  $\Phi (a * b) = \Phi a * \Phi b$ 
  proof cases
    case ab
      then have c-ab:
        coprime (a*b) d coprime a d coprime b d by simp+
      then have p1:  $[f (a * b) = a * b] \pmod d$  coprime (f (a * b)) n
        using m-prop[of a*b, OF c-ab(1)] by simp+
      moreover have p2:  $[f a = a] \pmod d$  coprime (f a) n
         $[f b = b] \pmod d$  coprime (f b) n
        using m-prop[of a, OF c-ab(2)]
          m-prop[of b, OF c-ab(3)] by simp+
      have p1s:  $[f (a * b) = (f a) * (f b)] \pmod d$ 
      proof -
        have  $[f (a * b) = a * b] \pmod d$ 
          using p1(1) by blast
        moreover have  $[a * b = f(a) * f(b)] \pmod d$ 
          using p2(1) p2(3) by (simp add: cong-mult cong-sym)
        ultimately show ?thesis using cong-trans by blast
      qed
  qed

```

```

have p2s: coprime (f a*f b) n
  using p2(2) p2(4) by simp
have  $\chi (f (a * b)) = \chi (f a * f b)$ 
  using p1s p2s p1(2) b by blast
then show ?thesis
  unfolding  $\Phi$ -def by (simp add: c-ab)
qed (simp-all add:  $\Phi$ -def)
qed
have d-gr-1:  $d > 1$  using assms(1,2)
  using  $\langle 0 < d \rangle$  by linarith
show  $\exists \Phi. dcharacter\ d\ \Phi \wedge (\forall n. \chi\ n = \Phi\ n * \chi_1\ n)$ 
proof (standard,rule conjI)
  show dcharacter  $d\ \Phi$ 
    unfolding dcharacter-def residues-nat-def dcharacter-axioms-def
    using d-gr-1  $\Phi$ -def f-def  $\Phi$ -mult  $\Phi$ -1  $\Phi$ -periodic by simp
  show  $\forall n. \chi\ n = \Phi\ n * \chi_1\ n$ 
proof
  fix k
  show  $\chi\ k = \Phi\ k * \chi_1\ k$ 
proof (cases coprime k n)
  case True
  then have coprime k d using assms(1) by auto
  then have  $\Phi(k) = \chi(f\ k)$  using  $\Phi$ -def by simp
  moreover have  $[f\ k = k] (mod\ d)$ 
    using m-prop[OF  $\langle coprime\ k\ d \rangle$ ] by simp
  moreover have  $\chi_1\ k = 1$ 
    using assms(3) principal-dchar-def  $\langle coprime\ k\ n \rangle$  by auto
  ultimately show  $\chi(k) = \Phi(k) * \chi_1(k)$ 
proof -
  assume  $\Phi\ k = \chi (f\ k) [f\ k = k] (mod\ d) \chi_1\ k = 1$ 
  then have  $\chi\ k = \chi (f\ k)$ 
    using  $\langle local.induced-modulus\ d \rangle induced-modulus-altdef1\ assms(1) \langle d > 0 \rangle$ 
    True  $\langle coprime\ k\ d \rangle m-prop(2)$  by auto
  also have  $\dots = \Phi\ k$  by (simp add:  $\langle \Phi\ k = \chi (f\ k) \rangle$ )
  also have  $\dots = \Phi\ k * \chi_1\ k$  by (simp add:  $\langle \chi_1\ k = 1 \rangle$ )
  finally show ?thesis by simp
qed
next
case False
hence  $\chi\ k = 0$ 
  using eq-zero-iff by blast
moreover have  $\chi_1\ k = 0$ 
  using False assms(3) principal-dchar-def by simp
ultimately show ?thesis by simp
qed
qed
qed
next
assume  $(\exists \Phi. dcharacter\ d\ \Phi \wedge (\forall k. \chi\ k = \Phi\ k * \chi_1\ k))$ 

```

then obtain Φ where 1: d character $d \Phi$ ($\forall k. \chi k = \Phi k * \chi_1 k$) by blast
show induced-modulus d
unfolding induced-modulus-def
proof (rule conjI,fact,safe)
fix k
assume 2: coprime $k n$ [$k = 1$] (mod d)
then have $\chi_1 k = 1 \Phi k = 1$
proof (simp add: assms(3) principal-dchar-def)
have $\Phi k = \Phi (k \bmod d)$ by (simp add: dcharacter.mod[OF 1(1), of k])
also have $\dots = \Phi (1 \bmod d)$ using cong-def[of $k 1 d$] 2(2) by simp
also have $\dots = \Phi 1$ using assms(2) 1(1) dcharacter.mod by blast
also have $\dots = 1$ using dcharacter.Suc-0[OF 1(1)] by simp
finally show $\Phi k = 1$ by simp
qed
then show $\chi k = 1$ using 1(2) by simp
qed
qed

7.4 The conductor of a character

context d character
begin

definition conductor = Min { d . induced-modulus d }

lemma conductor-fin: finite { d . induced-modulus d }

proof –
let $?A = \{d. \text{induced-modulus } d\}$
have $?A \subseteq \{d. d \text{ dvd } n\}$
unfolding induced-modulus-def by blast
moreover have finite { $d. d \text{ dvd } n\}$ using n by simp
ultimately show finite $?A$ using finite-subset by auto
qed

lemma conductor-induced: induced-modulus conductor

proof –
have { $d. \text{induced-modulus } d\} \neq \{\}$ using induced-modulus-modulus by blast
then show induced-modulus conductor
using Min-in[OF conductor-fin] conductor-def by auto
qed

lemma conductor-le-iff: conductor $\leq a \iff (\exists d \leq a. \text{induced-modulus } d)$

unfolding conductor-def using conductor-fin induced-modulus-modulus by (subst Min-le-iff) auto

lemma conductor-ge-iff: conductor $\geq a \iff (\forall d. \text{induced-modulus } d \longrightarrow d \geq a)$

unfolding conductor-def using conductor-fin induced-modulus-modulus by (subst Min-ge-iff) auto

lemma *conductor-leI*: $\text{induced-modulus } d \implies \text{conductor} \leq d$
by (*subst conductor-le-iff*) *auto*

lemma *conductor-geI*: $(\bigwedge d. \text{induced-modulus } d \implies d \geq a) \implies \text{conductor} \geq a$
by (*subst conductor-ge-iff*) *auto*

lemma *conductor-dvd*: $\text{conductor } \text{dvd } n$
using *conductor-induced* **unfolding** *induced-modulus-def* **by** *blast*

lemma *conductor-le-modulus*: $\text{conductor} \leq n$
using *conductor-dvd* **by** (*rule dvd-imp-le*) (*use n in auto*)

lemma *conductor-gr-0*: $\text{conductor} > 0$
unfolding *conductor-def* **using** *zero-not-ind-mod*
using *conductor-def* *conductor-induced* *neq0-conv* **by** *fastforce*

lemma *conductor-eq-1-iff-principal*: $\text{conductor} = 1 \iff \chi = \text{principal-dchar } n$
proof
assume $\text{conductor} = 1$
then have *induced-modulus 1*
using *conductor-induced* **by** *auto*
then show $\chi = \text{principal-dchar } n$
using *one-induced-iff-principal* **by** *blast*
next
assume $\chi = \text{principal-dchar } n$
then have *im-1: induced-modulus 1* **using** *one-induced-iff-principal* **by** *auto*
show $\text{conductor} = 1$
proof –
have $\text{conductor} \leq 1$
using *conductor-fin* *Min-le[OF conductor-fin,simplified,OF im-1]*
by (*simp add: conductor-def[symmetric]*)
then show *?thesis* **using** *conductor-gr-0* **by** *auto*
qed
qed

lemma *conductor-principal [simp]*: $\chi = \text{principal-dchar } n \implies \text{conductor} = 1$
by (*subst conductor-eq-1-iff-principal*)

lemma *nonprimitive-imp-conductor-less*:
assumes $\neg \text{primitive-dchar } n \chi$
shows $\text{conductor} < n$
proof –
obtain d **where** $d: \text{induced-modulus } d \text{ } d < n$
using *primitive-dchar-iff* *assms* **by** *blast*
from $d(1)$ **have** $\text{conductor} \leq d$
by (*rule conductor-leI*)
also have $\dots < n$ **by** *fact*
finally show *?thesis* .
qed

lemma (in *nonprimitive-dchar*) *conductor-less-modulus: conductor < n*
using *nonprimitive-imp-conductor-less nonprimitive* **by** *metis*

Theorem 8.18

theorem *primitive-principal-form:*

defines $\chi_1 \equiv$ *principal-dchar n*

assumes $\chi \neq$ *principal-dchar n*

shows $\exists \Phi.$ *primitive-dchar conductor* $\Phi \wedge (\forall n. \chi(n) = \Phi(n) * \chi_1(n))$

proof –

from *n* **have** *n-pos: n > 0* **by** *simp*

define *d* **where** $d =$ *conductor*

have *induced: induced-modulus d*

unfolding *d-def* **using** *conductor-induced* **by** *blast*

then **have** *d-not-1: d ≠ 1*

using *one-induced-iff-principal assms* **by** *auto*

hence *d-gt-1: d > 1* **using** *conductor-gr-0* **by** (*auto simp: d-def*)

from *induced* **obtain** Φ **where** Φ -*def: dcharacter d* $\Phi \wedge (\forall n. \chi n = \Phi n * \chi_1 n)$

using *d-not-1*

by (*subst (asm) induced-modulus-altdef2*) (*auto simp: d-def conductor-dvd χ_1 -def*)

have *phi-dchars: $\Phi \in$ dcharacters d* **using** Φ -*def dcharacters-def* **by** *auto*

interpret $\Phi:$ *dcharacter d residue-mult-group d* Φ

using Φ -*def* **by** *auto*

have Φ -*prim: primitive-dchar d* Φ

proof (*rule ccontr*)

assume \neg *primitive-dchar d* Φ

then **obtain** *q* **where**

1: q dvd d \wedge q < d \wedge Φ .induced-modulus q

unfolding Φ .*induced-modulus-def* Φ .*primitive-dchar-iff* **by** *blast*

then **have** *2: induced-modulus q*

proof –

{**fix** *k*

assume *mod-1: [k = 1] (mod q)*

assume *cop: coprime k n*

have $\chi(k) = \Phi(k) * \chi_1(k)$ **using** Φ -*def* **by** *auto*

also **have** $\dots = \Phi(k)$

using *cop* **by** (*simp add: assms principal-dchar-def*)

also **have** $\dots = 1$

using *1 mod-1 Φ .induced-modulus-def*

(induced-modulus d) cop induced-modulus-def **by** *auto*

finally **have** $\chi(k) = 1$ **by** *blast*}

then **show** *?thesis*

using *induced-modulus-def 1* (*induced-modulus d*) **by** *auto*
qed

from *1* **have** $q < d$ **by** *simp*
moreover have $d \leq q$ **unfolding** *d-def*
by (*intro conductor-leI*) *fact*
ultimately show *False* **by** *linarith*
qed

from Φ -*def* Φ -*prim* *d-def* *phi-dchars* **show** *?thesis* **by** *blast*
qed

definition *primitive-extension* :: *nat* \Rightarrow *complex* **where**
primitive-extension =
(*SOME* Φ . *primitive-dchar conductor* $\Phi \wedge (\forall k. \chi k = \Phi k * \text{principal-dchar } n$
k))

lemma
assumes *nonprincipal*: $\chi \neq \text{principal-dchar } n$
shows *primitive-primitive-extension*: *primitive-dchar conductor primitive-extension*
and *principal-decomposition*: $\chi k = \text{primitive-extension } k * \text{principal-dchar}$
n k
proof –
note $*$ = *someI-ex*[*OF primitive-principal-form*[*OF nonprincipal*], *folded primitive-extension-def*]
from $*$ **show** *primitive-dchar conductor primitive-extension* **by** *blast*
from $*$ **show** $\chi k = \text{primitive-extension } k * \text{principal-dchar } n k$ **by** *blast*
qed

end

7.5 The connection between primitivity and separability

lemma *residue-mult-group-coset*:
fixes *m n m1 m2* :: *nat* **and** *f* :: *nat* \Rightarrow *nat* **and** *G H*
defines *G* \equiv *residue-mult-group n*
defines *H* \equiv *residue-mult-group m*
defines *f* \equiv ($\lambda k. k \bmod m$)
assumes $b \in (\text{rcosets}_G \text{ kernel } G H f)$
assumes $m1 \in b$ $m2 \in b$
assumes $n > 1$ $m \text{ dvd } n$
shows $m1 \bmod m = m2 \bmod m$
proof –
have *h-1*: $1_H = 1$
using *assms(2)* **unfolding** *residue-mult-group-def totatives-def* **by** *simp*

from *assms(4)*
obtain *a* :: *nat* **where** *cos-expr*: $b = (\text{kernel } G H f) \#>_G a \wedge a \in \text{carrier } G$
using *RCOSETS-def*[*of G kernel G H f*] **by** *blast*

then have cop : $coprime\ a\ n$
using $assms(1)$ **unfolding** $residue-mult-group-def\ totatives-def$ **by** $auto$

obtain a' **where** $[a * a' = 1] \pmod n$
using $cong-solve-coprime-nat[OF\ cop]$ **by** $auto$
then have $a-inv$: $(a*a') \pmod n = 1$
using $cong-def[of\ a*a'\ 1\ n]$ $assms(7)$ **by** $simp$

have $m1 \in (\bigcup h \in kernel\ G\ H\ f. \{h \otimes_G a\})$
 $m2 \in (\bigcup h \in kernel\ G\ H\ f. \{h \otimes_G a\})$
using $r-coset-def[of\ G\ kernel\ G\ H\ f\ a]$ $cos-expr\ assms(5,6)$ **by** $blast+$
then have $m1 \in (\bigcup h \in kernel\ G\ H\ f. \{(h * a) \pmod n\})$
 $m2 \in (\bigcup h \in kernel\ G\ H\ f. \{(h * a) \pmod n\})$
using $assms(1)$ **unfolding** $residue-mult-group-def[of\ n]$ **by** $auto$
then obtain $m1'$ $m2'$ **where**
 $m-expr$: $m1 = (m1' * a) \pmod n \wedge m1' \in kernel\ G\ H\ f$
 $m2 = (m2' * a) \pmod n \wedge m2' \in kernel\ G\ H\ f$
by $blast$

have $eq-1$: $m1 \pmod m = a \pmod m$
proof –
have $m1 \pmod m = ((m1' * a) \pmod n) \pmod m$ **using** $m-expr$ **by** $blast$
also have $\dots = (m1' * a) \pmod m$
using $euclidean-semiring-cancel-class.mod-mod-cancel\ assms(8)$ **by** $blast$
also have $\dots = (m1' \pmod m) * (a \pmod m) \pmod m$
by $(simp\ add:\ mod-mult-eq)$
also have $\dots = (a \pmod m) \pmod m$
using $m-expr(1)\ h-1$ **unfolding** $kernel-def\ assms(3)$ **by** $simp$
also have $\dots = a \pmod m$ **by** $auto$
finally show $?thesis$ **by** $simp$
qed

have $eq-2$: $m2 \pmod m = a \pmod m$
proof –
have $m2 \pmod m = ((m2' * a) \pmod n) \pmod m$ **using** $m-expr$ **by** $blast$
also have $\dots = (m2' * a) \pmod m$
using $euclidean-semiring-cancel-class.mod-mod-cancel\ assms(8)$ **by** $blast$
also have $\dots = (m2' \pmod m) * (a \pmod m) \pmod m$
by $(simp\ add:\ mod-mult-eq)$
also have $\dots = (a \pmod m) \pmod m$
using $m-expr(2)\ h-1$ **unfolding** $kernel-def\ assms(3)$ **by** $simp$
also have $\dots = a \pmod m$ **by** $auto$
finally show $?thesis$ **by** $simp$
qed

from $eq-1\ eq-2$ **show** $?thesis$ **by** $argo$
qed

lemma $residue-mult-group-kernel-partition$:


```

fixes  $m\ n :: \text{nat}$  and  $f :: \text{nat} \Rightarrow \text{nat}$  and  $G\ H$ 
defines  $G \equiv \text{residue-mult-group } n$ 
defines  $H \equiv \text{residue-mult-group } m$ 
defines  $f \equiv (\lambda k. k \bmod m)$ 
assumes  $m > 1\ n > 0\ m \text{ dvd } n$ 
shows  $\text{partition } (\text{carrier } G)\ (\text{rcosets}_G\ \text{kernel } G\ H\ f)$ 
  and  $\text{card } (\text{rcosets}_G\ \text{kernel } G\ H\ f) = \text{totient } m$ 
  and  $\text{card } (\text{kernel } G\ H\ f) = \text{totient } n \text{ div totient } m$ 
  and  $b \in (\text{rcosets}_G\ \text{kernel } G\ H\ f) \implies b \neq \{\}$ 
  and  $b \in (\text{rcosets}_G\ \text{kernel } G\ H\ f) \implies \text{card } (\text{kernel } G\ H\ f) = \text{card } b$ 
  and  $\text{bij-betw } (\lambda b. (\text{the-elem } (f\ ' b)))\ (\text{rcosets}_G\ \text{kernel } G\ H\ f)\ (\text{carrier } H)$ 
proof –
  have  $1 < m$  by fact
  also have  $m \leq n$  using  $\langle n > 0 \rangle\ \langle m \text{ dvd } n \rangle$  by (intro dvd-imp-le) auto
  finally have  $n > 1$  .
  note  $mn = \langle m > 1 \rangle\ \langle n > 1 \rangle\ \langle m \text{ dvd } n \rangle\ \langle m \leq n \rangle$ 

  interpret  $n$ : residues-nat  $n\ G$ 
    using  $mn$  by unfold-locales (auto simp: assms)
  interpret  $m$ : residues-nat  $m\ H$ 
    using  $mn$  by unfold-locales (auto simp: assms)

  from  $mn$  have subset:  $f\ ' \text{carrier } G \subseteq \text{carrier } H$ 
    by (auto simp: assms(1–3) residue-mult-group-def totatives-def
      dest: coprime-common-divisor-nat intro!: Nat.gr0I)
  moreover have super-set:  $\text{carrier } H \subseteq f\ ' \text{carrier } G$ 
  proof safe
    fix  $k$  assume  $k \in \text{carrier } H$ 
    hence  $k: k > 0\ k \leq m$  coprime  $k\ m$ 
      by (auto simp: assms(2) residue-mult-group-def totatives-def)
    from  $mn\ \langle k \in \text{carrier } H \rangle$  have  $k < m$ 
      by (simp add: totatives-less assms(2) residue-mult-group-def)
    define  $P$  where  $P = \{p \in \text{prime-factors } n. \neg(p \text{ dvd } m)\}$ 
    define  $a$  where  $a = \prod P$ 
    have [simp]:  $a \neq 0$  by (auto simp: P-def a-def intro!: Nat.gr0I)
    have [simp]: prime-factors  $a = P$ 
    proof –
      have prime-factors  $a = \text{set-mset } (\text{sum prime-factorization } P)$ 
        unfolding a-def using  $mn$ 
        by (subst prime-factorization-prod)
        (auto simp: P-def prime-factors-dvd prime-gt-0-nat)
      also have sum prime-factorization  $P = (\sum p \in P. \{\#p\#})$ 
        using  $mn$  by (intro sum.cong) (auto simp: P-def prime-factorization-prime
prime-factors-dvd)
      finally show ?thesis by (simp add: P-def)
    qed

  from  $mn$  have coprime  $m\ a$ 
    by (subst coprime-iff-prime-factors-disjoint) (auto simp: P-def)

```

hence $\exists x. [x = k] \pmod{m} \wedge [x = 1] \pmod{a}$
by *(intro binary-chinese-remainder-nat)*
then obtain x **where** $x: [x = k] \pmod{m} [x = 1] \pmod{a}$
by *auto*
from $x(1)$ mn k **have** $[simp]: x \neq 0$
using *coprime-common-divisor[of k m]* **by** *(auto intro!: Nat.grOI simp: cong-def)*

from $x(2)$ **have** *coprime x a*
using *cong-imp-coprime cong-sym* **by** *force*
hence *coprime x (a * m)*
using k *cong-imp-coprime[OF cong-sym[OF x(1)]]* **by** *auto*
also have $?this \longleftrightarrow \text{coprime } x \ n$ **using** mn
by *(intro coprime-cong-prime-factors)*
(auto simp: prime-factors-product P-def in-prime-factors-iff)
finally have $x \pmod{n} \in \text{totatives } n$
using mn **by** *(auto simp: totatives-def intro!: Nat.grOI)*

moreover have $f (x \pmod{n}) = k$
using $x(1)$ k mn $(k < m)$ **by** *(auto simp: assms(3) cong-def mod-mod-cancel)*
ultimately show $k \in f \text{ ' carrier } G$
by *(auto simp: assms(1) residue-mult-group-def)*

qed

ultimately have *image-eq: f ' carrier G = carrier H* **by** *blast*

have $[simp]: f (k \otimes_G l) = f k \otimes_H f l$ **if** $k \in \text{carrier } G$ $l \in \text{carrier } G$ **for** k l
using *that mn* **by** *(auto simp: assms(1-3) residue-mult-group-def totatives-def mod-mod-cancel mod-mult-eq)*

interpret $f: \text{group-hom } G \ H$ f
using *subset* **by** *unfold-locales (auto simp: hom-def)*

show *bij-betw* $(\lambda b. (\text{the-elem } (f \text{ ' } b)))$ $(\text{rcosets}_G \text{ kernel } G \ H \ f)$ $(\text{carrier } H)$
unfolding *bij-betw-def*

proof

show *inj-on* $(\lambda b. (\text{the-elem } (f \text{ ' } b)))$ $(\text{rcosets}_G \text{ kernel } G \ H \ f)$
using $f.$ *FactGroup-inj-on* **unfolding** *FactGroup-def* **by** *auto*
have *eq: f ' carrier G = carrier H*
using *subset super-set* **by** *blast*
show $(\lambda b. \text{the-elem } (f \text{ ' } b)) \text{ ' } (\text{rcosets}_G \text{ kernel } G \ H \ f) = \text{carrier } H$
using $f.$ *FactGroup-onto[OF eq]* **unfolding** *FactGroup-def* **by** *simp*

qed

show *partition* $(\text{carrier } G)$ $(\text{rcosets}_G \text{ kernel } G \ H \ f)$

proof

show $\bigwedge a. a \in \text{carrier } G \implies$
 $\exists ! b. b \in \text{rcosets}_G \text{ kernel } G \ H \ f \wedge a \in b$

proof –

fix a

```

assume a-in:  $a \in \text{carrier } G$ 
show  $\exists ! b. b \in \text{rcosets}_G \text{ kernel } G \ H \ f \wedge a \in b$ 
proof –

  have  $\exists b. b \in \text{rcosets}_G \text{ kernel } G \ H \ f \wedge a \in b$ 
    using a-in n.rcosets-part-G[OF f.subgroup-kernel]
    by blast
  then show ?thesis
    using group.rcos-disjoint[OF n.is-group f.subgroup-kernel]
    by (auto simp: disjoint-def)
  qed
qed
next
  show  $\bigwedge b. b \in \text{rcosets}_G \text{ kernel } G \ H \ f \implies b \subseteq \text{carrier } G$ 
    using n.rcosets-part-G f.subgroup-kernel by auto
  qed

  have lagr:  $\text{card} (\text{carrier } G) = \text{card} (\text{rcosets}_G \text{ kernel } G \ H \ f) * \text{card} (\text{kernel } G \ H \ f)$ 
  using group.lagrange-finite[OF n.is-group n.fin f.subgroup-kernel] Coset.order-def[of G] by argo
  have k-size:  $\text{card} (\text{kernel } G \ H \ f) > 0$ 
    using f.subgroup-kernel finite-subset n.subgroupE(1) n.subgroupE(2) by fastforce
  have G-size:  $\text{card} (\text{carrier } G) = \text{totient } n$ 
    using n.order Coset.order-def[of G] by simp
  have H-size:  $\text{totient } m = \text{card} (\text{carrier } H)$ 
    using n.order Coset.order-def[of H] by simp
  also have  $\dots = \text{card} (\text{carrier } (G \ \text{Mod} \ \text{kernel } G \ H \ f))$ 
    using f.FactGroup-iso[OF image-eq] card-image f.FactGroup-inj-on f.FactGroup-onto image-eq by fastforce
  also have  $\dots = \text{card} (\text{carrier } G) \ \text{div} \ \text{card} (\text{kernel } G \ H \ f)$ 
  proof –
    have  $\text{card} (\text{carrier } (G \ \text{Mod} \ \text{kernel } G \ H \ f)) = \text{card} (\text{rcosets}_G \text{ kernel } G \ H \ f)$ 
      unfolding FactGroup-def by simp
    also have  $\dots = \text{card} (\text{carrier } G) \ \text{div} \ \text{card} (\text{kernel } G \ H \ f)$ 
      by (simp add: lagr k-size)
    finally show ?thesis by blast
  qed
  also have  $\dots = \text{totient } n \ \text{div} \ \text{card} (\text{kernel } G \ H \ f)$ 
    using G-size by argo
  finally have eq:  $\text{totient } m = \text{totient } n \ \text{div} \ \text{card} (\text{kernel } G \ H \ f)$  by simp
  show  $\text{card} (\text{kernel } G \ H \ f) = \text{totient } n \ \text{div} \ \text{totient } m$ 
  proof –
    have  $\text{totient } m \neq 0$ 
      using totient-0-iff[of m] assms(4) by blast
    have  $\text{card} (\text{kernel } G \ H \ f) \ \text{dvd} \ \text{totient } n$ 
      using lagr ( $\text{card} (\text{carrier } G) = \text{totient } n$ ) by auto

```

```

have totient m * card (kernel G H f) = totient n
  unfolding eq using ⟨card (kernel G H f) dvd totient n⟩ by auto
have totient n div totient m = totient m * card (kernel G H f) div totient m
  using ⟨totient m * card (kernel G H f) = totient n⟩ by auto
also have ... = card (kernel G H f)
  using nonzero-mult-div-cancel-left[OF ⟨totient m ≠ 0⟩] by blast
finally show ?thesis by auto
qed

show card (rcosets_G kernel G H f) = totient m
proof -
  have H-size: totient m = card (carrier H)
    using n.order Coset.order-def[of H] by simp
  also have ... = card (carrier (G Mod kernel G H f))
    using f.FactGroup-iso[OF image-eq] card-image f.FactGroup-inj-on f.FactGroup-onto
image-eq by fastforce
  also have card (carrier (G Mod kernel G H f)) =
    card (rcosets_G kernel G H f)
    unfolding FactGroup-def by simp
  finally show card (rcosets_G kernel G H f) = totient m
    by argo
qed

assume b ∈ rcosets_G kernel G H f
then show b ≠ {}
proof -
  have card b = card (kernel G H f)
    using ⟨b ∈ rcosets_G kernel G H f⟩ f.subgroup-kernel n.card-rcosets-equal
n.subgroupE(1) by auto
  then have card b > 0
    by (simp add: k-size)
  then show ?thesis by auto
qed

assume b-cos: b ∈ rcosets_G kernel G H f
show card (kernel G H f) = card b
  using group.card-rcosets-equal[OF n.is-group b-cos]
f.subgroup-kernel subgroup.subset by blast
qed

lemma primitive-iff-separable-lemma:
  assumes prod: (∀ n. χ n = Φ n * χ₁ n) ∧ primitive-dchar d Φ
  assumes ⟨d > 1⟩ ⟨0 < k⟩ ⟨d dvd k⟩ ⟨k > 1⟩
  shows (∑ m | m ∈ {1..k} ∧ coprime m k. Φ(m) * unity-root d m) =
    (totient k div totient d) * (∑ m | m ∈ {1..d} ∧ coprime m d. Φ(m) *
unity-root d m)
proof -
  from assms interpret Φ: primitive-dchar d residue-mult-group d Φ

```

```

    by auto
  define G where G = residue-mult-group k
  define H where H = residue-mult-group d
  define f where f = (λt. t mod d)

  from residue-mult-group-kernel-partition(2)[OF ⟨d > 1⟩ ⟨0 < k⟩ ⟨d dvd k⟩]
  have fin-cosets: finite (rcosets_G kernel G H f)
    using ⟨1 < d⟩ card.infinite by (fastforce simp: G-def H-def f-def)

  have fin-G: finite (carrier G)
    unfolding G-def residue-mult-group-def by simp

  have eq: (∑ m | m ∈ {1..k} ∧ coprime m k. Φ(m) * unity-root d m) =
    (∑ m | m ∈ carrier G . Φ(m) * unity-root d m)
    unfolding residue-mult-group-def totatives-def G-def
    by (rule sum.cong, auto)
  also have ... = sum (λm. Φ(m) * unity-root d m) (carrier G) by simp
  also have eq': ... = sum (sum (λm. Φ m * unity-root d (int m))) (rcosets_G
kernel G H f)
    by (rule disjoint-sum [symmetric])
    (use fin-G fin-cosets residue-mult-group-kernel-partition(1)[OF ⟨d > 1⟩ ⟨k >
0⟩ ⟨d dvd k⟩] in
    ⟨auto simp: G-def H-def f-def⟩)
  also have ... =
    (∑ b ∈ (rcosets_G kernel G H f) . (∑ m ∈ b. Φ m * unity-root d (int m))) by
simp
  finally have 1: (∑ m | m ∈ {1..k} ∧ coprime m k. Φ(m) * unity-root d m) =
    (∑ b ∈ (rcosets_G kernel G H f) . (∑ m ∈ b. Φ m * unity-root d
(int m)))
    using eq eq' by auto
  have eq''': ... =
    (∑ b ∈ (rcosets_G kernel G H f) . (totient k div totient d) * (Φ (the-lem (f ‘
b)) * unity-root d (int (the-lem (f ‘ b))))))
  proof (rule sum.cong, simp)
    fix b
    assume b-in: b ∈ (rcosets_G kernel G H f)
    note b-not-empty = residue-mult-group-kernel-partition(4)
      [OF ⟨d > 1⟩ ⟨0 < k⟩ ⟨d dvd k⟩ b-in[unfolded G-def H-def f-def]]
    {
      fix m1 m2
      assume m-in: m1 ∈ b m2 ∈ b
      have m-mod: m1 mod d = m2 mod d
        using residue-mult-group-coset[OF b-in[unfolded G-def H-def f-def] m-in ⟨k
> 1⟩ ⟨d dvd k⟩]
        by blast
    } note m-mod = this
    {
      fix m1 m2

```

```

assume m-in:  $m1 \in b$   $m2 \in b$ 
have  $\Phi$   $m1 * \text{unity-root } d$   $(\text{int } m1) = \Phi$   $m2 * \text{unity-root } d$   $(\text{int } m2)$ 
proof –
  have  $\Phi$ -periodic: periodic-arithmetic  $\Phi$   $d$  using  $\Phi$ .dir-periodic-arithmetic by
blast
  have 1:  $\Phi$   $m1 = \Phi$   $m2$ 
    using mod-periodic-arithmetic[OF periodic-arithmetic  $\Phi$   $d$ ] m-mod[OF
m-in] by simp
  have 2: unity-root  $d$   $m1 = \text{unity-root } d$   $m2$ 
    using m-mod[OF m-in] by (intro unity-root-cong) (auto simp: cong-def
simp flip: zmod-int)
  from 1 2 show ?thesis by simp
qed
} note all-eq-in-coset = this

from all-eq-in-coset b-not-empty
obtain  $l$  where l-prop:  $l \in b \wedge (\forall y \in b. \Phi$   $y * \text{unity-root } d$   $(\text{int } y) =$ 
 $\Phi$   $l * \text{unity-root } d$   $(\text{int } l))$  by blast

have  $(\sum m \in b. \Phi$   $m * \text{unity-root } d$   $(\text{int } m)) =$ 
 $((\text{totient } k \text{ div totient } d) * (\Phi$   $l * \text{unity-root } d$   $(\text{int } l)))$ 
proof –
  have  $(\sum m \in b. \Phi$   $m * \text{unity-root } d$   $(\text{int } m)) =$ 
 $(\sum m \in b. \Phi$   $l * \text{unity-root } d$   $(\text{int } l))$ 
  by (rule sum.cong,simp) (use all-eq-in-coset l-prop in blast)
  also have  $\dots = \text{card } b * \Phi$   $l * \text{unity-root } d$   $(\text{int } l)$ 
  by simp
  also have  $\dots = (\text{totient } k \text{ div totient } d) * \Phi$   $l * \text{unity-root } d$   $(\text{int } l)$ 
  using residue-mult-group-kernel-partition(3)[OF  $\langle d > 1 \rangle \langle 0 < k \rangle \langle d \text{ dvd } k \rangle$ ]
residue-mult-group-kernel-partition(5)
  [OF  $\langle d > 1 \rangle \langle 0 < k \rangle \langle d \text{ dvd } k \rangle$  b-in [unfolded G-def H-def f-def]]
  by argo
  finally have 2:
   $(\sum m \in b. \Phi$   $m * \text{unity-root } d$   $(\text{int } m)) =$ 
 $(\text{totient } k \text{ div totient } d) * \Phi$   $l * \text{unity-root } d$   $(\text{int } l)$ 
  by blast
  from b-not-empty 2 show ?thesis by auto
qed
also have  $\dots = ((\text{totient } k \text{ div totient } d) * (\Phi$   $(\text{the-elem } (f \text{ ‘ } b)) * \text{unity-root } d$ 
 $(\text{int } (\text{the-elem } (f \text{ ‘ } b))))))$ 
proof –
  have foral:  $(\bigwedge y. y \in b \implies f$   $y = f$   $l)$ 
  using m-mod l-prop unfolding f-def by blast
  have eq: the-elem  $(f \text{ ‘ } b) = f$   $l$ 
  using the-elem-image-unique[of - f  $l$ , OF b-not-empty foral] by simp
  have per: periodic-arithmetic  $\Phi$   $d$  using prod  $\Phi$ .dir-periodic-arithmetic by
blast
show ?thesis
  unfolding eq using mod-periodic-arithmetic[OF per, of  $l \text{ mod } d$   $l$ ]

```

by (*auto simp: f-def unity-root-mod zmod-int*)
qed
finally show $(\sum m \in b. \Phi m * \text{unity-root } d \text{ (int } m)) =$
 $((\text{totient } k \text{ div totient } d) * (\Phi (\text{the-elem } (f ' b)) * \text{unity-root } d \text{ (int$
 $(\text{the-elem } (f ' b))))))$
by blast
qed
have ... =
 $(\sum b \in (\text{rcosets}_G \text{ kernel } G H f) . (\text{totient } k \text{ div totient } d) * (\Phi (\text{the-elem}$
 $(f ' b)) * \text{unity-root } d \text{ (int } (\text{the-elem } (f ' b))))))$
by blast
also have eq'' :
 $\dots = (\sum h \in \text{carrier } H . (\text{totient } k \text{ div totient } d) * (\Phi (h) * \text{unity-root } d \text{ (int$
 $(h))))$
unfolding $H\text{-def } G\text{-def } f\text{-def}$
by (*rule sum.reindex-bij-betw[OF residue-mult-group-kernel-partition(6)[OF <d > 1> <0 <k> <d dvd k>]]*)
finally have 2 : $(\sum m \mid m \in \{1..k\} \wedge \text{coprime } m k. \Phi(m) * \text{unity-root } d m) =$
 $(\text{totient } k \text{ div totient } d) * (\sum h \in \text{carrier } H . (\Phi (h) * \text{unity-root } d$
 $(\text{int } (h))))$
using 1 **by** (*simp add: eq'' eq''' sum-distrib-left*)
also have ... = $(\text{totient } k \text{ div totient } d) * (\sum m \mid m \in \{1..d\} \wedge \text{coprime } m d . (\Phi$
 $(m) * \text{unity-root } d \text{ (int } (m))))$
unfolding $H\text{-def residue-mult-group-def}$ **by** (*simp add: totatives-def Suc-le-eq*)
finally show *?thesis* **by simp**
qed

Theorem 8.19

theorem (*in dcharacter*) *primitive-iff-separable*:

primitive-dchar $n \chi \iff (\forall k > 0. \text{separable } k)$

proof (*cases* $\chi = \text{principal-dchar } n$)

case *True*

thus *?thesis*

using *principal-not-primitive principal-not-totally-separable* **by auto**

next

case *False*

note *nonprincipal = this*

show *?thesis*

proof

assume *primitive-dchar* $n \chi$

then interpret *A: primitive-dchar* n *residue-mult-group* $n \chi$ **by auto**

show $(\forall k. k > 0 \implies \text{separable } k)$

using $n.A.\text{primitive-encoding}(2)$ **by blast**

next

assume *tot-separable*: $\forall k > 0. \text{separable } k$

{

assume *as*: $\neg \text{primitive-dchar } n \chi$

have $\exists r. r \neq 0 \wedge \neg \text{coprime } r n \wedge \text{gauss-sum } r \neq 0$

proof –

```

from  $n$  have  $n > 0$  by simp
define  $d$  where  $d = \text{conductor}$ 
have  $d > 0$  unfolding  $d\text{-def}$  using conductor-gr-0 .
then have  $d > 1$  using nonprincipal d-def conductor-eq-1-iff-principal by
auto
have  $d < n$  unfolding  $d\text{-def}$  using nonprimitive-imp-conductor-less[OF as]
.
have  $d \text{ dvd } n$  unfolding  $d\text{-def}$  using conductor-dvd by blast
define  $r$  where  $r = n \text{ div } d$ 
have  $0: r \neq 0$  unfolding  $r\text{-def}$ 
using  $\langle 0 < n \rangle \langle d \text{ dvd } n \rangle \text{ dvd-div-gt0}$  by auto
have  $\text{gcd } r \ n > 1$ 
unfolding  $r\text{-def}$ 
proof –
have  $n \text{ div } d > 1$  using  $\langle 1 < n \rangle \langle d < n \rangle \langle d \text{ dvd } n \rangle$  by auto
have  $n \text{ div } d \text{ dvd } n$  using  $\langle d \text{ dvd } n \rangle$  by force
have  $\text{gcd } (n \text{ div } d) \ n = n \text{ div } d$  using gcd-nat.absorb1[OF  $\langle n \text{ div } d \text{ dvd } n \rangle$ ]
by blast
then show  $1 < \text{gcd } (n \text{ div } d) \ n$  using  $\langle n \text{ div } d > 1 \rangle$  by argo
qed
then have  $1: \neg \text{coprime } r \ n$  by auto
define  $\chi_1$  where  $\chi_1 = \text{principal-dchar } n$ 
from primitive-principal-form[OF nonprincipal]
obtain  $\Phi$  where
prod:  $(\forall k. \chi(k) = \Phi(k) * \chi_1(k)) \wedge \text{primitive-dchar } d \ \Phi$ 
using  $d\text{-def}$  unfolding  $\chi_1\text{-def}$  by blast
then have prod1:  $(\forall k. \chi(k) = \Phi(k) * \chi_1(k)) \text{ primitive-dchar } d \ \Phi$  by blast+
then interpret  $\Phi$ : primitive-dchar d residue-mult-group d  $\Phi$ 
by auto

have gauss-sum  $r = (\sum m = 1..n . \chi(m) * \text{unity-root } n \ (m*r))$ 
unfolding gauss-sum-def by blast
also have  $\dots = (\sum m = 1..n . \Phi(m) * \chi_1(m) * \text{unity-root } n \ (m*r))$ 
by (rule sum.cong,auto simp add: prod)
also have  $\dots = (\sum m \mid m \in \{1..n\} \wedge \text{coprime } m \ n. \Phi(m) * \chi_1(m) * \text{unity-root}$ 
 $n \ (m*r))$ 
by (intro sum.mono-neutral-right) (auto simp: \chi_1-def principal-dchar-def)
also have  $\dots = (\sum m \mid m \in \{1..n\} \wedge \text{coprime } m \ n. \Phi(m) * \chi_1(m) * \text{unity-root}$ 
 $d \ m)$ 
proof (rule sum.cong,simp)
fix  $x$ 
assume  $x \in \{m \in \{1..n\}. \text{coprime } m \ n\}$ 
have  $\text{unity-root } n \ (\text{int } (x * r)) = \text{unity-root } d \ (\text{int } x)$ 
using unity-div-num[OF  $\langle n > 0 \rangle \langle d > 0 \rangle \langle d \text{ dvd } n \rangle$ ]
by (simp add: algebra-simps r-def)
then show  $\Phi \ x * \chi_1 \ x * \text{unity-root } n \ (\text{int } (x * r)) =$ 
 $\Phi \ x * \chi_1 \ x * \text{unity-root } d \ (\text{int } x)$  by auto
qed
also have  $\dots = (\sum m \mid m \in \{1..n\} \wedge \text{coprime } m \ n. \Phi(m) * \text{unity-root } d \ m)$ 

```



```

    by (rule sum.cong,auto simp add:  $\chi_1$ -def principal-dchar-def)
    also have ... = (totient n div totient d) * ( $\sum m \mid m \in \{1..d\} \wedge \text{coprime } m$ 
d.  $\Phi(m) * \text{unity-root } d$  m)
    using primitive-iff-separable-lemma[OF prod <d > 1> <n > 0> <d dvd n> <n
> 1>] by blast
    also have ... = (totient n div totient d) *  $\Phi$ .gauss-sum 1
    proof -
      have  $\Phi$ .gauss-sum 1 = ( $\sum m = 1..d . \Phi m * \text{unity-root } d$  (int (m )))
      by (simp add:  $\Phi$ .gauss-sum-def)
      also have ... = ( $\sum m \mid m \in \{1..d\} . \Phi m * \text{unity-root } d$  (int m))
      by (rule sum.cong,auto)
      also have ... = ( $\sum m \mid m \in \{1..d\} \wedge \text{coprime } m$  d.  $\Phi(m) * \text{unity-root } d$ 
m)
      by (rule sum.mono-neutral-right) (use  $\Phi$ .eq-zero in auto)
      finally have  $\Phi$ .gauss-sum 1 = ( $\sum m \mid m \in \{1..d\} \wedge \text{coprime } m$  d.  $\Phi(m)
* \text{unity-root } d$  m)
      by blast
      then show ?thesis by metis
    qed
    finally have g-expr: gauss-sum r = (totient n div totient d) *  $\Phi$ .gauss-sum
1
    by blast
    have t-non-0: totient n div totient d  $\neq$  0
    by (simp add: <0 < n> <d dvd n> dvd-div-gt0 totient-dvd)
    have (norm ( $\Phi$ .gauss-sum 1))2 = d
    using  $\Phi$ .primitive-encoding(3) by simp
    then have  $\Phi$ .gauss-sum 1  $\neq$  0
    using <0 < d> by auto
    then have 2: gauss-sum r  $\neq$  0
    using g-expr t-non-0 by auto
    from 0 1 2 show  $\exists r. r \neq 0 \wedge \neg \text{coprime } r n \wedge \text{gauss-sum } r \neq 0$ 
    by blast
  qed
}
note contr = this

show primitive-dchar n  $\chi$ 
proof (rule ccontr)
  assume  $\neg$  primitive-dchar n  $\chi$ 
  then obtain r where 1: r  $\neq$  0  $\wedge \neg \text{coprime } r n \wedge \text{gauss-sum } r \neq 0$ 
  using contr by blast
  from global-separability-condition tot-separable
  have 2: ( $\forall k > 0. \neg \text{coprime } k n \longrightarrow \text{gauss-sum } k = 0$ )
  by blast
  from 1 2 show False by blast
qed
qed
qed

```

Theorem 8.20

theorem (in *primitive-dchar*) *fourier-primitive*:
includes *no-vec-lambda-notation*
fixes $\tau :: \text{complex}$
defines $\tau \equiv \text{gauss-sum } 1 / \text{sqr}t \ n$
shows $\chi \ m = \tau / \text{sqr}t \ n * (\sum k=1..n. \text{cnj} (\chi \ k) * \text{unity-root } n \ (-m*k))$
and $\text{norm } \tau = 1$
proof –
have *chi-not-principal*: $\chi \neq \text{principal-dchar } n$
using *principal-not-totally-separable primitive-encoding(2)* **by** *blast*

then have *case-0*: $(\sum k=1..n. \chi \ k) = 0$
proof –
have $\text{sum } \chi \ \{0..n-1\} = \text{sum } \chi \ \{1..n\}$
using *periodic-arithmetic-sum-periodic-arithmetic-shift[OF dir-periodic-arithmetic, of 1]* n
by *auto*
also have $\{0..n-1\} = \{..<n\}$
using n **by** *auto*
finally show $(\sum n = 1..n . \chi \ n) = 0$
using *sum-dcharacter-block chi-not-principal* **by** *simp*
qed

have $\chi \ m =$
 $(\sum k = 1..n. 1 / \text{of-nat } n * \text{gauss-sum-int } (- \text{int } k) * \text{unity-root } n \ (\text{int } (m * k)))$
using *dcharacter-fourier-expansion[of m]* **by** *auto*
also have $\dots = (\sum k = 1..n. 1 / \text{of-nat } n * \text{gauss-sum } (\text{nat } ((- k) \text{ mod } n)) * \text{unity-root } n \ (\text{int } (m * k)))$
by (*auto simp: gauss-sum-int-conv-gauss-sum*)
also have $\dots = (\sum k = 1..n. 1 / \text{of-nat } n * \text{cnj} (\chi \ (\text{nat } ((- k) \text{ mod } n))) * \text{gauss-sum } 1 * \text{unity-root } n \ (\text{int } (m * k)))$
proof (*rule sum.cong,simp*)
fix k
assume $k \in \{1..n\}$
have $\text{gauss-sum } (\text{nat } (- \text{int } k \text{ mod } \text{int } n)) = \text{cnj} (\chi \ (\text{nat } (- \text{int } k \text{ mod } \text{int } n))) * \text{gauss-sum } 1$
proof (*cases nat ((- k) mod n) > 0*)
case *True*
then show *?thesis*
using *mp[OF spec[OF primitive-encoding(2)] True]*
unfolding *separable-def* **by** *auto*
next
case *False*
then have *nat-0*: $\text{nat } ((- k) \text{ mod } n) = 0$ **by** *blast*
show *?thesis*
proof –
have $\text{gauss-sum } (\text{nat } (- \text{int } k \text{ mod } \text{int } n)) = \text{gauss-sum } 0$
using *nat-0* **by** *argo*
also have $\dots = (\sum m = 1..n. \chi \ m)$

```

    unfolding gauss-sum-def by (rule sum.cong) auto
  also have ... = 0 using case-0 by blast
  finally have 1: gauss-sum (nat (- int k mod int n)) = 0
    by blast

  have 2: cnj (χ (nat (- int k mod int n))) = 0
    using nat-0 zero-eq-0 by simp
  show ?thesis using 1 2 by simp
qed
qed
then show 1 / of-nat n * gauss-sum (nat (- int k mod int n)) * unity-root n
(int (m * k)) =
  1 / of-nat n * cnj (χ (nat (- int k mod int n))) * gauss-sum 1 *
unity-root n (int (m * k))
  by auto
qed
also have ... = (∑ k = 1..n. 1 / of-nat n * cnj (χ (nat (- int k mod int n))) *
gauss-sum 1 * unity-root n (int (m * (nat (int k mod int n))))))
proof (rule sum.cong,simp)
  fix x
  assume x ∈ {1..n}
  have unity-root n (m * x) = unity-root n (m * x mod n)
    using unity-root-mod-nat[of n m*x] by (simp add: nat-mod-as-int)
  also have ... = unity-root n (m * (x mod n))
    by (rule unity-root-cong)
    (auto simp: cong-def mod-mult-right-eq simp flip: zmod-int of-nat-mult)
  finally have unity-root n (m * x) = unity-root n (m * (x mod n)) by blast
  then show 1 / of-nat n * cnj (χ (nat (- int x mod int n))) *
gauss-sum 1 * unity-root n (int (m * x)) =
  1 / of-nat n * cnj (χ (nat (- int x mod int n))) * gauss-sum 1 *
unity-root n (int (m * nat (int x mod int n)))
    by (simp add: nat-mod-as-int)
qed
also have ... = (∑ k = 0..n-1. 1 / of-nat n * cnj (χ k) * gauss-sum 1 *
unity-root n (- int (m * k)))
proof -
  have b: bij-betw (λk. nat((-k) mod n)) {1..n} {0..n-1}
    unfolding bij-betw-def
  proof
    show inj-on (λk. nat (- int k mod int n)) {1..n}
      unfolding inj-on-def
    proof (safe)
      fix x y
      assume a1: x ∈ {1..n} y ∈ {1..n}
      assume a2: nat (- x mod n) = nat (- y mod n)
      then have (- x) mod n = - y mod n
        using n eq-nat-nat-iff by auto
      then have [-int x = - int y] (mod n)
        using cong-def by blast
    end
  end

```

```

then have [x = y] (mod n)
  by (simp add: cong-int-iff cong-minus-minus-iff)
then have cong: x mod n = y mod n using cong-def by blast
then show x = y
proof (cases x = n)
  case True then show ?thesis using cong a1(2) by auto
next
  case False
  then have x mod n = x using a1(1) by auto
  then have y ≠ n using a1(1) local.cong by fastforce
  then have y mod n = y using a1(2) by auto
  then show ?thesis using ⟨x mod n = x⟩ cong by linarith
qed
qed
show (λk. nat (- int k mod int n)) ‘ {1..n} = {0..n - 1}
  unfolding image-def
proof
  let ?A = {y. ∃ x ∈ {1..n}. y = nat (- int x mod int n)}
  let ?B = {0..n - 1}
  show ?A ⊆ ?B
  proof
    fix y
    assume y ∈ {y. ∃ x ∈ {1..n}. y = nat (- int x mod int n)}
    then obtain x where x ∈ {1..n} ∧ y = nat (- int x mod int n) by blast
    then show y ∈ {0..n - 1} by (simp add: nat-le-iff of-nat-diff)
  qed
  show ?A ⊇ ?B
  proof
    fix x
    assume 1: x ∈ {0..n-1}
    then have n - x ∈ {1..n}
      using n by auto
    have x = nat (- int (n-x) mod int n)
    proof -
      have nat (- int (n-x) mod int n) = nat (int x) mod int n
        apply (simp add: int-ops(6), rule conjI)
        using ⟨n - x ∈ {1..n}⟩ by force+
      also have ... = x
        using 1 n by auto
      finally show ?thesis by presburger
    qed
    then show x ∈ {y. ∃ x ∈ {1..n}. y = nat (- int x mod int n)}
      using ⟨n - x ∈ {1..n}⟩ by blast
  qed
qed
qed
show ?thesis
proof -
  have 1: (∑ k = 1..n. 1 / of-nat n * cnj (χ (nat (- int k mod int n)))) *

```

```

gauss-sum 1 * unity-root n (int (m * nat (int k mod int n)))) =
  (∑ x = 1..n. 1 / of-nat n * cnj (χ (nat (- int x mod int n)))) *
gauss-sum 1 * unity-root n (- int (m * nat (- int x mod int n))))
proof (rule sum.cong,simp)
fix x
have (int m * (int x mod int n)) mod n = (m*x) mod n
  by (simp add: mod-mult-right-eq zmod-int)
also have ... = (- ((- int (m*x) mod n))) mod n
  by (simp add: mod-minus-eq of-nat-mod)
  have (int m * (int x mod int n)) mod n = (- (int m * (- int x mod int
n))) mod n
  apply(subst mod-mult-right-eq,subst add.inverse-inverse[symmetric],subst
(5) add.inverse-inverse[symmetric])
  by (subst minus-mult-minus,subst mod-mult-right-eq[symmetric],auto)
then have unity-root n (int m * (int x mod int n)) =
  unity-root n (- (int m * (- int x mod int n)))
  using unity-root-mod[of n int m * (int x mod int n)]
  unity-root-mod[of n - (int m * (- int x mod int n))] by argo
then show 1 / of-nat n * cnj (χ (nat (- int x mod int n))) *
gauss-sum 1 *
unity-root n (int (m * nat (int x mod int n))) =
1 / of-nat n * cnj (χ (nat (- int x mod int n))) *
gauss-sum 1 *
unity-root n (- int (m * nat (- int x mod int n))) by auto
qed
also have 2: (∑ x = 1..n. 1 / of-nat n * cnj (χ (nat (- int x mod int n))) *
gauss-sum 1 * unity-root n (- int (m * nat (- int x mod int n)))) =
(∑ md = 0..n - 1. 1 / of-nat n * cnj (χ md) * gauss-sum 1 *
unity-root n (- int (m * md)))
  using sum.reindex-bij-betw[OF b, of λmd. 1 / of-nat n * cnj (χ md) *
gauss-sum 1 * unity-root n (- int (m * md))]
  by blast
also have 3: ... = (∑ k = 0..n - 1.
1 / of-nat n * cnj (χ k) * gauss-sum 1 *
unity-root n (- int (m * k))) by blast
finally have (∑ k = 1..n. 1 / of-nat n * cnj (χ (nat (- int k mod int n))) *
gauss-sum 1 * unity-root n (int (m * nat (int k mod int n)))) =
(∑ k = 0..n - 1.
1 / of-nat n * cnj (χ k) * gauss-sum 1 *
unity-root n (- int (m * k))) using 1 2 3 by argo
then show ?thesis by blast
qed
qed
also have ... = (∑ k = 1..n.
1 / of-nat n * cnj (χ k) * gauss-sum 1 *
unity-root n (- int (m * k)))
proof -
let ?f = (λk. 1 / of-nat n * cnj (χ k) * gauss-sum 1 * unity-root n (- int (m
* k)))

```

```

have ?f 0 = 0
  using zero-eq-0 by auto
have ?f n = 0
  using zero-eq-0 mod-periodic-arithmetic[OF dir-periodic-arithmetic, of n 0]
  by simp
have ( $\sum n = 0..n - 1. ?f n$ ) = ( $\sum n = 1..n - 1. ?f n$ )
  using sum-shift-lb-Suc0-0[of ?f, OF ‹?f 0 = 0›]
  by auto
also have ... = ( $\sum n = 1..n. ?f n$ )
proof (rule sum.mono-neutral-left,simp,simp,safe)
  fix i
  assume  $i \in \{1..n\}$   $i \notin \{1..n - 1\}$ 
  then have  $i = n$  using n by auto
  then show  $1 / \text{of-nat } n * \text{cnj } (\chi i) * \text{gauss-sum } 1 * \text{unity-root } n (- \text{int } (m$ 
*  $i)) = 0$ 
    using ‹?f n = 0› by blast
  qed
finally show ?thesis by blast
qed
also have ... = ( $\sum k = 1..n. (\tau / \text{sqrt } n) * \text{cnj } (\chi k) * \text{unity-root } n (- \text{int } (m$ 
*  $k)))$ 
proof (rule sum.cong,simp)
  fix x
  assume  $x \in \{1..n\}$ 
  have  $\tau / \text{sqrt } (\text{real } n) = 1 / \text{of-nat } n * \text{gauss-sum } 1$ 
  proof -
    have  $\tau / \text{sqrt } (\text{real } n) = \text{gauss-sum } 1 / \text{sqrt } n / \text{sqrt } n$ 
    using assms by auto
    also have ... =  $\text{gauss-sum } 1 / (\text{sqrt } n * \text{sqrt } n)$ 
    by (subst divide-divide-eq-left,subst of-real-mult,blast)
    also have ... =  $\text{gauss-sum } 1 / n$ 
    using real-sqrt-mult-self by simp
    finally show ?thesis by simp
  qed
  then show
     $1 / \text{of-nat } n * \text{cnj } (\chi x) * \text{gauss-sum } 1 * \text{unity-root } n (- \text{int } (m * x)) =$ 
     $(\tau / \text{sqrt } n) * \text{cnj } (\chi x) * \text{unity-root } n (- \text{int } (m * x))$  by simp
  qed
  also have ... =  $\tau / \text{sqrt } (\text{real } n) *$ 
    ( $\sum k = 1..n. \text{cnj } (\chi k) * \text{unity-root } n (- \text{int } (m * k))$ )
  proof -
    have ( $\sum k = 1..n. \tau / \text{sqrt } (\text{real } n) * \text{cnj } (\chi k) * \text{unity-root } n (- \text{int } (m * k))$ )
    =
      ( $\sum k = 1..n. \tau / \text{sqrt } (\text{real } n) * (\text{cnj } (\chi k) * \text{unity-root } n (- \text{int } (m *$ 
 $k)))$ )
    by (rule sum.cong,simp, simp add: algebra-simps)
    also have ... =  $\tau / \text{sqrt } (\text{real } n) * (\sum k = 1..n. \text{cnj } (\chi k) * \text{unity-root } n (-$ 
 $\text{int } (m * k)))$ 
    by (rule sum-distrib-left[symmetric])

```

```

finally show ?thesis by blast
qed

finally show  $\chi m = (\tau / \text{sqrt } (\text{real } n)) * (\sum_{k=1..n} \text{cnj } (\chi k) * \text{unity-root } n (- \text{int } m * \text{int } k))$  by simp

have 1:  $\text{norm } (\text{gauss-sum } 1) = \text{sqrt } n$ 
using gauss-sum-1-mod-square-eq-k[OF primitive-encoding(2)]
by (simp add: cmod-def)
from assms have 2:  $\text{norm } \tau = \text{norm } (\text{gauss-sum } 1) / |\text{sqrt } n|$ 
by (simp add: norm-divide)
show  $\text{norm } \tau = 1$  using 1 2 by simp
qed

unbundle vec-lambda-notation

end

```

8 The Pólya–Vinogradov Inequality

```

theory Polyva-Vinogradov
imports
  Gauss-Sums
  Dirichlet-Series.Divisor-Count
begin

unbundle no-vec-lambda-notation

```

8.1 The case of primitive characters

We first prove a stronger variant of the Pólya–Vinogradov inequality for primitive characters. The fully general variant will then simply be a corollary of this. First, we need some bounds on logarithms, exponentials, and the harmonic numbers:

```

lemma ln-add-one-self-less-self:
  fixes  $x :: \text{real}$ 
  assumes  $x > 0$ 
  shows  $\ln (1 + x) < x$ 
proof –
  have  $0 \leq x$   $0 < x$   $\exp x > 0$   $1+x > 0$  using assms by simp+
  have  $1 + x < 1 + x + x^2 / 2$ 
    using  $\langle 0 < x \rangle$  by auto
  also have  $\dots \leq \exp x$ 
    using exp-lower-Taylor-quadratic[OF  $\langle 0 \leq x \rangle$ ] by blast
  finally have  $1 + x < \exp (x)$  by blast
  then have  $\ln (1 + x) < \ln (\exp (x))$ 
    using ln-less-cancel-iff[OF  $\langle 1+x > 0 \rangle$   $\langle \exp(x) > 0 \rangle$ ] by auto
  also have  $\dots = x$  using ln-exp by blast

```

finally show *?thesis* by *auto*
qed

lemma *exp-1-bounds*:

assumes $x > (0::real)$

shows $\exp 1 > (1 + 1 / x) \text{ powr } x$ and $\exp 1 < (1 + 1 / x) \text{ powr } (x+1)$

proof –

have $\ln (1 + 1 / x) < 1 / x$

using *ln-add-one-self-less-self* *assms* by *simp*

thus $\exp 1 > (1 + 1 / x) \text{ powr } x$ using *assms*

by (*simp add: field-simps powr-def*)

next

have $1 < (x + 1) * \ln ((x + 1) / x)$ (*is - < ?f x*)

proof (*rule DERIV-neg-imp-decreasing-at-top*[*where ?f = ?f*])

fix *t* assume *t*: $x \leq t$

have (*?f has-field-derivative* ($\ln (1 + 1 / t) - 1 / t$)) (*at t*)

using *t assms* by (*auto intro!: derivative-eq-intros simp:divide-simps*)

moreover have $\ln (1 + 1 / t) - 1 / t < 0$

using *ln-add-one-self-less-self*[*of 1 / t*] *t assms* by *auto*

ultimately show $\exists y. ((\lambda t. (t + 1) * \ln ((t + 1) / t)) \text{ has-real-derivative } y)$

(*at t*) $\wedge y < 0$

by *blast*

qed *real-asymp*

thus $\exp 1 < (1 + 1 / x) \text{ powr } (x + 1)$

using *assms* by (*simp add: powr-def field-simps*)

qed

lemma *harm-aux-ineq-1*:

fixes $k :: real$

assumes $k > 1$

shows $1 / k < \ln (1 + 1 / (k - 1))$

proof –

have $k-1 > 0$ ($k > 0$) using *assms* by *simp+*

from *exp-1-bounds(2)*[*OF* ($k-1 > 0$)]

have $\exp 1 < (1 + 1 / (k - 1)) \text{ powr } k$ by *simp*

then have *n-z*: $(1 + 1 / (k - 1)) \text{ powr } k > 0$

using *assms not-exp-less-zero* by *auto*

have $(1::real) = \ln (\exp(1))$ using *ln-exp* by *auto*

also have $\dots < \ln ((1 + 1 / (k - 1)) \text{ powr } k)$

using *ln-less-cancel-iff*[*of exp(1),simplified,OF* ($(1 + 1 / (k - 1)) \text{ powr } k > 0$)]

exp-1-bounds[*OF* ($k - 1 > 0$)] by *simp*

also have $\dots = k * \ln (1 + 1 / (k - 1))$

using *ln-powr n-z* by *simp*

finally have $1 < k * \ln (1 + 1 / (k - 1))$

by *blast*

then show *?thesis* using *assms* by (*simp add: field-simps*)

qed

lemma *harm-aux-ineq-2-lemma*:
assumes $x \geq (0::\text{real})$
shows $1 < (x + 1) * \ln (1 + 2 / (2 * x + 1))$
proof –
have $0 < \ln (1+2/(2*x+1)) - 1 / (x + 1)$ (**is** $- < ?f x$)
proof (*rule DERIV-neg-imp-decreasing-at-top*[**where** $?f = ?f$])
fix t **assume** $t: x \leq t$
from *assms* t **have** $3 + 8 * t + 4 * t^2 > 0$
by (*intro add-pos-nonneg*) *auto*
hence $*$: $3 + 8 * t + 4 * t^2 \neq 0$
by *auto*
have ($?f$ *has-field-derivative* $(-1 / ((1 + t)^2 * (3 + 8 * t + 4 * t^2)))$) (*at* t)
apply (*insert assms* $t *$, (*rule derivative-eq-intros refl | simp add: add-pos-pos*)+)
apply (*auto simp: divide-simps*)
apply (*auto simp: algebra-simps power2-eq-square*)
done
moreover **have** $-1 / ((1 + t)^2 * (3 + 8 * t + 4 * t^2)) < 0$
using t *assms* **by** (*intro divide-neg-pos mult-pos-pos add-pos-nonneg*) *auto*
ultimately show $\exists y. (?f \text{ has-real-derivative } y) (\text{at } t) \wedge y < 0$
by *blast*
qed *real-asymp*
thus $1 < (x + 1) * \ln (1+2/(2*x+1))$
using *assms* **by** (*simp add: field-simps*)
qed

lemma *harm-aux-ineq-2*:
fixes $k :: \text{real}$
assumes $k \geq 1$
shows $1 / (k + 1) < \ln (1 + 2 / (2 * k + 1))$
proof –
have $k > 0$ **using** *assms* **by** *auto*
have $1 < (k + 1) * \ln (1 + 2 / (2 * k + 1))$
using *harm-aux-ineq-2-lemma* *assms* **by** *simp*
then show $?thesis$
by (*simp add: <0 < k> add-pos-pos mult.commute mult-imp-div-pos-less*)
qed

lemma *nat-0-1-induct* [*case-names 0 1 step*]:
assumes $P 0 P 1 \wedge n. n \geq 1 \implies P n \implies P (\text{Suc } n)$
shows $P n$
proof (*induction n rule: less-induct*)
case (*less n*)
show $?case$
using *assms*(3)[*OF - less.IH*[*of n - 1*]]
by (*cases n ≤ 1*)
(insert assms(1-2), auto simp: eval-nat-numeral le-Suc-eq)
qed

```

lemma harm-less-ln:
  fixes m :: nat
  assumes m > 0
  shows harm m < ln (2 * m + 1)
  using assms
proof (induct m rule: nat-0-1-induct)
  case 0
  then show ?case by blast
next
  case 1
  have harm 1 = (1::real) unfolding harm-def by simp
  have harm 1 < ln (3::real)
    by (subst (harm 1 = 1,subst ln3-gt-1,simp))
  then show ?case by simp
next
  case (step n)
  have harm (n+1) = harm n + 1/(n+1)
    by ((subst Suc-eq-plus1[symmetric])+,subst harm-Suc,subst inverse-eq-divide,blast)
  also have ... < ln (real (2 * n + 1)) + 1/(n+1)
    using step(1-2) by auto
  also have ... < ln (real (2 * n + 1)) + ln (1+2/(2*n+1))
  proof -
    from step(1) have real n ≥ 1 by simp
    have 1 / real (n + 1) < ln (1 + 2 / real (2 * n + 1))
      using harm-aux-ineq-2[OF (1 ≤ (real n))] by (simp add: add.commute)
    then show ?thesis by auto
  qed
  also have ... = ln ((2 * n + 1) * (1+2/(2*n+1)))
    by (rule ln-mult[symmetric],simp,simp add: field-simps)
  also have ... = ln (2*(n+1)+1)
  proof -
    have (2 * n + 1) * (1+2/(2*n+1)) = 2*(n+1)+1
      by (simp add: field-simps)
    then show ?thesis by presburger
  qed
  finally show ?case by simp
qed

```

Theorem 8.21

```

theorem (in primitive-dchar) polya-vinogradov-inequality-primitive:
  fixes x :: nat
  shows norm (∑ m=1..x. χ m) < sqrt n * ln n
proof -
  define τ :: complex where τ = gauss-sum 1 div sqrt n
  have τ-mod: norm τ = 1 using fourier-primitive(2)
    by (simp add: τ-def)
  {
    fix m
    have χ m = (τ div sqrt n) * (∑ k = 1..n. (cnj (χ k)) * unity-root n (-m*k))
  }

```

using *fourier-primitive(1)[of m] τ -def by blast*
note *chi-expr = this*
have $(\sum m = 1..x. \chi(m)) = (\sum m = 1..x. (\tau \text{ div } \text{sqrt } n) * (\sum k = 1..n. (\text{cnj } (\chi k)) * \text{unity-root } n (-m*k)))$
by (*rule sum.cong[OF refl] (use chi-expr in blast)*)
also have $\dots = (\sum m = 1..x. (\sum k = 1..n. (\tau \text{ div } \text{sqrt } n) * ((\text{cnj } (\chi k)) * \text{unity-root } n (-m*k))))$
by (*rule sum.cong,simp,simp add: sum-distrib-left*)
also have $\dots = (\sum k = 1..n. (\sum m = 1..x. (\tau \text{ div } \text{sqrt } n) * ((\text{cnj } (\chi k)) * \text{unity-root } n (-m*k))))$
by (*rule sum.swap*)
also have $\dots = (\sum k = 1..n. (\tau \text{ div } \text{sqrt } n) * (\text{cnj } (\chi k) * (\sum m = 1..x. \text{unity-root } n (-m*k))))$
by (*rule sum.cong,simp,simp add: sum-distrib-left*)
also have $\dots = (\sum k = 1..<n. (\tau \text{ div } \text{sqrt } n) * (\text{cnj } (\chi k) * (\sum m = 1..x. \text{unity-root } n (-m*k))))$
using *n by (intro sum.mono-neutral-right) (auto intro: eq-zero)*
also have $\dots = (\tau \text{ div } \text{sqrt } n) * (\sum k = 1..<n. (\text{cnj } (\chi k) * (\sum m = 1..x. \text{unity-root } n (-m*k))))$
by (*simp add: sum-distrib-left*)
finally have $(\sum m = 1..x. \chi(m)) = (\tau \text{ div } \text{sqrt } n) * (\sum k = 1..<n. (\text{cnj } (\chi k) * (\sum m = 1..x. \text{unity-root } n (-m*k))))$
by *blast*
hence *eq: sqrt n * ($\sum m=1..x. \chi(m)$) = $\tau * (\sum k=1..<n. (\text{cnj } (\chi k) * (\sum m=1..x. \text{unity-root } n (-m*k)))$)*
by *auto*
define *f where* $f = (\lambda k. (\sum m = 1..x. \text{unity-root } n (-m*k)))$

hence $(\text{sqrt } n) * \text{norm}(\sum m = 1..x. \chi(m)) = \text{norm}(\tau * (\sum k=1..<n. (\text{cnj } (\chi k) * (\sum m = 1..x. \text{unity-root } n (-m*k)))))$
proof –
have $\text{norm}(\text{sqrt } n * (\sum m=1..x. \chi(m))) = \text{norm } (\text{sqrt } n) * \text{norm}((\sum m = 1..x. \chi(m)))$
by (*simp add: norm-mult*)
also have $\dots = (\text{sqrt } n) * \text{norm}((\sum m = 1..x. \chi(m)))$
by *simp*
finally have *1: norm((sqrt n) * ($\sum m = 1..x. \chi(m)$)) = (sqrt n) * norm(($\sum m = 1..x. \chi(m)$))*
by *blast*
then show *?thesis using eq by algebra*
qed
also have $\dots = \text{norm } (\sum k = 1..<n. (\text{cnj } (\chi k) * (\sum m = 1..x. \text{unity-root } n (-m*k))))$
by (*simp add: norm-mult τ -mod*)
also have $\dots \leq (\sum k = 1..<n. \text{norm } (\text{cnj } (\chi k) * (\sum m = 1..x. \text{unity-root } n (-m*k))))$
using *norm-sum by blast*
also have $\dots = (\sum k = 1..<n. \text{norm } (\text{cnj } (\chi k)) * \text{norm}((\sum m = 1..x. \text{unity-root } n (-m*k))))$

```

    by (rule sum.cong,simp, simp add: norm-mult)
  also have ... ≤ (∑ k = 1..<n. norm((∑ m = 1..x. unity-root n (-m*k))))
  proof -
    show ?thesis
    proof (rule sum-mono)
      fix k
      assume k ∈ {1..<n}
      define sum-aux :: real where sum-aux = norm (∑ m=1..x. unity-root n (-
int m * int k))
      have sum-aux ≥ 0 unfolding sum-aux-def by auto
      have norm (cnj (χ k)) ≤ 1 using norm-le-1[of k] by simp
      then have norm (cnj (χ k)) * sum-aux ≤ 1 * sum-aux
        using (sum-aux ≥ 0) by (simp add: mult-left-le-one-le)
      then show norm (cnj (χ k)) *
        norm (∑ m = 1..x. unity-root n (- int m * int k))
        ≤ norm (∑ m = 1..x. unity-root n (- int m * int k))
        unfolding sum-aux-def by argo
    qed
  qed
  also have ... = (∑ k = 1..<n. norm(f k))
    using f-def by blast
  finally have 24: (sqrt n) * norm(∑ m = 1..x. χ(m)) ≤ (∑ k = 1..<n. norm(f
k))
    by blast

{
  fix k :: int
  have f(n-k) = cnj(f(k))
  proof -
    have f(n-k) = (∑ m = 1..x. unity-root n (-m*(n-k)))
      unfolding f-def by blast
    also have ... = (∑ m = 1..x. unity-root n (m*k))
    proof (rule sum.cong,simp)
      fix xa
      assume xa ∈ {1..x}
      have (k * int xa - int n * int xa) mod int n = (k * int xa - 0) mod int n
        by (intro mod-diff-cong) auto
      thus unity-root n (-int xa * (int n - k)) = unity-root n (int xa * k)
        unfolding ring-distrib by (intro unity-root-cong) (auto simp: cong-def
algebra-simps)
    qed
  also have ... = cnj(f(k))
  proof -
    have cnj(f(k)) = cnj (∑ m = 1..x. unity-root n (- int m * k))
      unfolding f-def by blast
    also have cnj (∑ m = 1..x. unity-root n (- int m * k)) =
      (∑ m = 1..x. cnj(unity-root n (- int m * k)))
      by (rule cnj-sum)
    also have ... = (∑ m = 1..x. unity-root n (int m * k))

```

```

      by (intro sum.cong) (auto simp: unity-root-uminus)
      finally show ?thesis by auto
    qed
    finally show  $f(n-k) = cnj(f(k))$  by blast
  qed
  hence  $norm(f(n-k)) = norm(cnj(f(k)))$  by simp
  hence  $norm(f(n-k)) = norm(f(k))$  by auto
}
note eq = this
have 25:
  odd  $n \implies (\sum k = 1..n - 1. norm (f (int k))) \leq$ 
     $2 * (\sum k = 1..(n-1) div 2. norm (f (int k)))$ 
  even  $n \implies (\sum k = 1..n - 1. norm (f (int k))) \leq$ 
     $2 * (\sum k = 1..(n-2) div 2. norm (f (int k))) + norm(f(n div 2))$ 
proof -
  assume odd  $n$ 
  define  $g$  where  $g = (\lambda k. norm (f k))$ 
  have  $(n-1) div 2 = n div 2$  using ‹odd  $n$ ›  $n$ 
    using div-mult-self1-is-m[OF pos2, of  $n-1$ ]
    odd-two-times-div-two-nat[OF ‹odd  $n$ ›] by linarith
  have  $(\sum i=1..n-1. g i) = (\sum i \in \{1..n div 2\} \cup \{n div 2 <..n-1\}. g i)$ 
    using  $n$  by (intro sum.cong, auto)
  also have  $\dots = (\sum i \in \{1..n div 2\}. g i) + (\sum i \in \{n div 2 <..n-1\}. g i)$ 
    by (subst sum.union-disjoint, auto)
  also have  $(\sum i \in \{n div 2 <..n-1\}. g i) = (\sum i \in \{1..n - (n div 2 + 1)\}. g (n$ 
-  $i))$ 
    by (rule sum.reindex-bij-witness[of -  $\lambda i. n - i \lambda i. n - i$ ], auto)
  also have  $\dots \leq (\sum i \in \{1..n div 2\}. g (n - i))$ 
    by (intro sum-mono2, simp, auto simp add: g-def)
  finally have 1:  $(\sum i=1..n-1. g i) \leq (\sum i=1..n div 2. g i + g (n - i))$ 
    by (simp add: sum.distrib)
  have  $(\sum i=1..n div 2. g i + g (n - i)) = (\sum i=1..n div 2. 2 * g i)$ 
    unfolding g-def
    apply (rule sum.cong, simp)
    using eq int-ops(6) by force
  also have  $\dots = 2 * (\sum i=1..n div 2. g i)$ 
    by (rule sum-distrib-left[symmetric])
  finally have 2:  $(\sum i=1..n div 2. g i + g (n - i)) = 2 * (\sum i=1..n div 2. g i)$ 
    by blast
  from 1 2 have  $(\sum i=1..n-1. g i) \leq 2 * (\sum i=1..n div 2. g i)$  by algebra
  then show  $(\sum n = 1..n - 1. norm (f (int n))) \leq 2 * (\sum n = 1..(n-1) div$ 
2.  $norm (f (int n)))$ 
    unfolding g-def ‹ $(n-1) div 2 = n div 2$ › by blast
next
  assume even  $n$ 
  define  $g$  where  $g = (\lambda n. norm (f (n)))$ 
  have  $(n-2) div 2 = n div 2 - 1$  using ‹even  $n$ ›  $n$  by simp
  have  $(\sum i=1..n-1. g i) = (\sum i \in \{1..<n div 2\} \cup \{n div 2\} \cup \{n div 2 <..n-1\}.$ 
 $g i)$ 

```

```

    using n by (intro sum.cong,auto)
    also have ... = ( $\sum_{i \in \{1..<n \text{ div } 2\}} g i$ ) + ( $\sum_{i \in \{n \text{ div } 2 <..n-1\}} g i$ ) +
g(n div 2)
    by (subst sum.union-disjoint,auto)
    also have ( $\sum_{i \in \{n \text{ div } 2 <..n-1\}} g i$ ) = ( $\sum_{i \in \{1..n - (n \text{ div } 2 + 1)\}} g (n -$ 
i))
    by (rule sum.reindex-bij-witness[of -  $\lambda i. n - i$   $\lambda i. n - i$ ],auto)
    also have ...  $\leq$  ( $\sum_{i \in \{1..<n \text{ div } 2\}} g (n - i)$ )
    proof (intro sum-mono2,simp)
      have  $n - n \text{ div } 2 = n \text{ div } 2$  using  $\langle$ even  $n$  $\rangle$  n by auto
      then have  $n - (n \text{ div } 2 + 1) < n \text{ div } 2$ 
        using n by (simp add: divide-simps)
      then show  $\{1..n - (n \text{ div } 2 + 1)\} \subseteq \{1..<n \text{ div } 2\}$  by fastforce
    qed auto
    finally have 1: ( $\sum_{i=1..n-1} g i$ )  $\leq$  ( $\sum_{i=1..<n \text{ div } 2} g i + g (n - i)$ ) +
g(n div 2)
    by (simp add: sum.distrib)
    have ( $\sum_{i=1..<n \text{ div } 2} g i + g (n - i)$ ) = ( $\sum_{i=1..<n \text{ div } 2} 2 * g i$ )
      unfolding g-def
      apply (rule sum.cong,simp)
      using eq int-ops(6) by force
    also have ... =  $2 * (\sum_{i=1..<n \text{ div } 2} g i)$ 
      by (rule sum-distrib-left[symmetric])
    finally have 2: ( $\sum_{i=1..<n \text{ div } 2} g i + g (n - i)$ ) =  $2 * (\sum_{i=1..<n \text{ div } 2}$ 
g i)
      by blast
    from 1 2 have 3: ( $\sum_{i=1..n-1} g i$ )  $\leq$   $2 * (\sum_{i=1..<n \text{ div } 2} g i) + g(n \text{ div}$ 
2) by algebra
    then have ( $\sum_{i=1..n-1} g i$ )  $\leq$   $2 * (\sum_{i=1..(n-2) \text{ div } 2} g i) + g(n \text{ div } 2)$ 
    proof -
      have  $\{1..<n \text{ div } 2\} = \{1..(n-2) \text{ div } 2\}$  by auto
      then have ( $\sum_{i=1..<n \text{ div } 2} g i$ ) = ( $\sum_{i=1..(n-2) \text{ div } 2} g i$ )
        by (rule sum.cong,simp)
      then show ?thesis using 3 by presburger
    qed
    then show ( $\sum_{k=1..n-1} \text{norm } (f (int k))$ )  $\leq$   $2 * (\sum_{n=1..(n-2) \text{ div}}$ 
2.  $\text{norm } (f (int n))$ ) +  $g(n \text{ div } 2)$ 
      unfolding g-def by blast
    qed

```

```

{fix k :: int
assume 1  $\leq$  k  $k \leq n \text{ div } 2$ 
have  $k \leq n - 1$ 
  using  $\langle$ k  $\leq n \text{ div } 2$  $\rangle$  n by linarith
define y where  $y = \text{unity-root } n (-k)$ 
define z where  $z = \text{exp } (-(\text{pi} * k / n) * i)$ 
have  $z^2 = \text{exp } (2 * (-(\text{pi} * k / n) * i))$ 
  unfolding z-def using exp-double[symmetric] by blast

```

also have $\dots = y$
unfolding $y\text{-def}$ $unity\text{-root}\text{-conv}\text{-exp}$ **by** ($simp$ add : $algebra\text{-simps}$)
finally have $z\text{-eq}$: $y = z^2$ **by** $blast$
have $z\text{-not}\text{-}0$: $z \neq 0$
using $z\text{-eq}$ **by** ($simp$ add : $z\text{-def}$)

then have $y \neq 1$
using $unity\text{-root}\text{-eq}\text{-}1\text{-iff}\text{-int}$ $\langle 1 \leq k \rangle$ $\langle k \leq n - 1 \rangle$ $not\text{-less}$
 $unity\text{-root}\text{-eq}\text{-}1\text{-iff}\text{-int}$ $y\text{-def}$ $z\text{dvd}\text{-not}\text{-zless}$ **by** $auto$

have $f(k) = (\sum m = 1..x . y^m)$
unfolding $f\text{-def}$ $y\text{-def}$
by ($subst$ $unity\text{-root}\text{-pow}$, $rule$ $sum.cong$, $simp$, $simp$ add : $algebra\text{-simps}$)
also have sum : $\dots = (\sum m = 1..<x+1 . y^m)$
by ($rule$ $sum.cong$, $fastforce$, $simp$)
also have $\dots = (\sum m = 0..<x+1 . y^m) - 1$
by ($subst$ (2) $sum.atLeast\text{-Suc}\text{-lessThan}$) $auto$
also have $\dots = (y^{x+1} - 1) \text{div} (y - 1) - 1$
using $geometric\text{-sum}[OF \langle y \neq 1 \rangle, \text{of } x+1]$ **by** ($simp$ add : $atLeast0LessThan$)

also have $\dots = (y^{x+1} - 1 - (y-1)) \text{div} (y - 1)$
proof –
have $y - 1 \neq 0$ **using** $\langle y \neq 1 \rangle$ **by** $simp$
show $?thesis$
using $divide\text{-diff}\text{-eq}\text{-iff}$ $[OF \langle y - 1 \neq 0 \rangle, \text{of } (y^{x+1} - 1) 1]$ **by** $auto$
qed

also have $\dots = (y^{x+1} - y) \text{div} (y - 1)$
by ($simp$ add : $algebra\text{-simps}$)
also have $\dots = y * (y^x - 1) \text{div} (y - 1)$
by ($simp$ add : $algebra\text{-simps}$)
also have $\dots = z^2 * ((z^2)^x - 1) \text{div} (z^2 - 1)$
unfolding $z\text{-eq}$ **by** $blast$
also have $\dots = z^2 * (z^{2*x} - 1) \text{div} (z^2 - 1)$
by ($subst$ $power\text{-mult}[symmetric, \text{of } z^2 x]$, $blast$)
also have $\dots = z^{x+1} * ((z^x - inverse(z^x))) / (z - inverse(z))$
proof –
have $z^x \neq 0$ **using** $z\text{-not}\text{-}0$ **by** $auto$
have 1 : $z^{2*x} - 1 = z^x * (z^x - inverse(z^x))$
by ($simp$ add : $semiring\text{-normalization}\text{-rules}(36)$ $right\text{-inverse}[OF \langle z^x \neq 0 \rangle]$
 $right\text{-diff}\text{-distrib}'$)
have 2 : $z^2 - 1 = z * (z - inverse(z))$
by ($simp$ add : $right\text{-diff}\text{-distrib}'$ $semiring\text{-normalization}\text{-rules}(29)$ $right\text{-inverse}[OF \langle z \neq 0 \rangle]$)

have 3 : $z^2 * (z^x / z) = z^{x+1}$
proof –
have $z^2 * (z^x / z) = z^2 * (z^x * inverse z)$
by ($simp$ add : $inverse\text{-eq}\text{-divide}$)
also have $\dots = z^{x+1}$

by (simp add: algebra-simps power2-eq-square right-inverse[OF ‹ $z \neq 0$ ›])
 finally show ?thesis by blast
 qed
 have $z^2 * (z^{2 * x} - 1) / (z^2 - 1) =$
 $z^2 * (z^{\widehat{x}} * (z^{\widehat{x}} - \text{inverse}(z^{\widehat{x}}))) / (z * (z - \text{inverse}(z)))$
 by (subst 1, subst 2, blast)
 also have ... = $(z^2 * (z^{\widehat{x}} / z)) * ((z^{\widehat{x}} - \text{inverse}(z^{\widehat{x}})) / (z - \text{inverse}(z)))$
 by simp
 also have ... = $z^{\widehat{x+1}} * ((z^{\widehat{x}} - \text{inverse}(z^{\widehat{x}})) / (z - \text{inverse}(z)))$
 by (subst 3, simp)
 finally show ?thesis by simp
 qed
 finally have $f(k) = z^{\widehat{x+1}} * ((z^{\widehat{x}} - \text{inverse}(z^{\widehat{x}})) / (z - \text{inverse}(z)))$ by
 blast

then have $\text{norm}(f(k)) = \text{norm}(z^{\widehat{x+1}} * (((z^{\widehat{x}} - \text{inverse}(z^{\widehat{x}})) / (z - \text{inverse}(z))))$ by auto
 also have ... = $\text{norm}(z^{\widehat{x+1}}) * \text{norm}(((z^{\widehat{x}} - \text{inverse}(z^{\widehat{x}})) / (z - \text{inverse}(z))))$

using norm-mult by blast
 also have ... = $\text{norm}(((z^{\widehat{x}} - \text{inverse}(z^{\widehat{x}})) / (z - \text{inverse}(z))))$
 proof -
 have $\text{norm}(z) = 1$
 unfolding z-def by auto
 have $\text{norm}(z^{\widehat{x+1}}) = 1$
 by (subst norm-power, simp add: ‹ $\text{norm}(z) = 1$ ›)
 then show ?thesis by simp

qed
 also have ... = $\text{norm}((\exp(-(x * \pi * k / n) * i) - \exp((x * \pi * k / n) * i)) \text{div}$
 $(\exp(-(\pi * k / n) * i) - \exp((\pi * k / n) * i)))$

proof -
 have 1: $z^{\widehat{x}} = \exp(-(x * \pi * k / n) * i)$
 unfolding z-def
 by (subst exp-of-nat-mult[symmetric], simp add: algebra-simps)
 have $\text{inverse}(z^{\widehat{x}}) = \text{inverse}(\exp(-(x * \pi * k / n) * i))$
 using ‹ $z^{\widehat{x}} = \exp(-(x * \pi * k / n) * i)$ › by auto
 also have ... = $(\exp((x * \pi * k / n) * i))$
 by (simp add: exp-minus)
 finally have 2: $\text{inverse}(z^{\widehat{x}}) = \exp((x * \pi * k / n) * i)$ by simp
 have 3: $\text{inverse } z = \exp((\pi * k / n) * i)$
 by (simp add: exp-minus z-def)
 show ?thesis using 1 2 3 z-def by simp

qed
 also have ... = $\text{norm}((\sin(x * \pi * k / n)) \text{div} (\sin(\pi * k / n)))$

proof -
 have num: $(\exp(-(x * \pi * k / n) * i) - \exp((x * \pi * k / n) * i)) = (-2 * i * \sin((x * \pi * k / n)))$

proof -

have 1: $\exp(-(x\pi k/n)*i) = \cos(-(x\pi k/n)) + i * \sin(-(x\pi k/n))$
 $\exp((x\pi k/n)*i) = \cos((x\pi k/n)) + i * \sin((x\pi k/n))$
using Euler Im-complex-of-real Im-divide-of-nat Im-i-times Re-complex-of-real
complex-Re-of-int complex-i-mult-minus exp-zero mult.assoc
mult.commute **by** force+
have $(\exp(-(x\pi k/n)*i) - \exp((x\pi k/n)*i)) =$
 $(\cos(-(x\pi k/n)) + i * \sin(-(x\pi k/n))) -$
 $(\cos((x\pi k/n)) + i * \sin((x\pi k/n)))$
using 1 **by** argo
also have ... = $-2*i * \sin((x\pi k/n))$ **by** simp
finally show ?thesis **by** blast
qed

have den: $(\exp(-(pi*k/n)*i) - \exp((pi*k/n)*i)) = -2*i * \sin((pi*k/n))$
proof -
have 1: $\exp(-(pi*k/n)*i) = \cos(-(pi*k/n)) + i * \sin(-(pi*k/n))$
 $\exp((pi*k/n)*i) = \cos((pi*k/n)) + i * \sin((pi*k/n))$
using Euler Im-complex-of-real Im-divide-of-nat Im-i-times Re-complex-of-real
complex-Re-of-int complex-i-mult-minus exp-zero mult.assoc
mult.commute **by** force+
have $(\exp(-(pi*k/n)*i) - \exp((pi*k/n)*i)) =$
 $(\cos(-(pi*k/n)) + i * \sin(-(pi*k/n))) -$
 $(\cos((pi*k/n)) + i * \sin((pi*k/n)))$
using 1 **by** argo
also have ... = $-2*i * \sin((pi*k/n))$ **by** simp
finally show ?thesis **by** blast
qed

have norm($(\exp(-(x\pi k/n)*i) - \exp((x\pi k/n)*i)) \text{ div}$
 $(\exp(-(pi*k/n)*i) - \exp((pi*k/n)*i)) =$
 $\text{norm}((-2*i * \sin((x\pi k/n))) \text{ div } (-2*i * \sin((pi*k/n))))$
using num den **by** presburger
also have ... = $\text{norm}(\sin((x\pi k/n)) \text{ div } \sin((pi*k/n)))$
by (simp add: norm-divide)
finally show ?thesis **by** blast
qed

also have ... = $\text{norm}(\sin(x\pi k/n)) \text{ div } \text{norm}(\sin(pi*k/n))$
by (simp add: norm-divide)
also have ... $\leq 1 \text{ div } \text{norm}(\sin(pi*k/n))$
proof -
have $\text{norm}(\sin(pi*k/n)) \geq 0$ **by** simp
have $\text{norm}(\sin(x\pi k/n)) \leq 1$ **by** simp
then show ?thesis
using divide-right-mono[OF $\langle \text{norm}(\sin(x\pi k/n)) \leq 1 \rangle \langle \text{norm}(\sin(pi*k/n)) \geq 0 \rangle$]
by blast
qed

finally have 26: $\text{norm}(f(k)) \leq 1 \text{ div } \text{norm}(\sin(pi*k/n))$

```

by blast

{
  fix t
  assume  $t \geq 0 \wedge t \leq \pi \operatorname{div} 2$ 
  then have  $t \in \{0.. \pi \operatorname{div} 2\}$  by auto
  have convex-on  $\{0.. \pi/2\}$   $(\lambda x. -\sin x)$ 
    by (rule convex-on-realI[where  $f' = \lambda x. -\cos x$ ])
      (auto intro!: derivative-eq-intros simp: cos-monotone-0- $\pi$ -le)
  from convex-onD-Icc[OF this  $\langle t \in \{0.. \pi \operatorname{div} 2\} \rangle$ ] have  $\sin(t) \geq (2 \operatorname{div} \pi) * t$ 
by simp
}
note sin-ineq = this

have sin-ineq-inst:  $\sin((\pi * k) / n) \geq (2 * k) / n$ 
proof -
  have  $\pi / n \geq 0$  by simp
  have 1:  $(\pi * k) / n \geq 0$  using  $\langle 1 \leq k \rangle$  by auto
  have  $(\pi * k) / n = (\pi / n) * k$  by simp
  also have  $\dots \leq (\pi / n) * (n / 2)$ 
    using mult-left-mono[of  $k \ n \ 2 \ \pi / n$ ]
       $\langle k \leq n \operatorname{div} 2 \rangle \langle 0 \leq \pi / n \rangle$  by linarith
  also have  $\dots \leq \pi / 2$ 
    by (simp add: divide-simps)
  finally have 2:  $(\pi * k) / n \leq \pi / 2$  by auto

  have  $(2 / \pi) * (\pi * k / n) \leq \sin((\pi * k) / n)$ 
    using sin-ineq[OF 1 2] by blast
  then show  $\sin((\pi * k) / n) \geq (2 * k) / n$ 
    by auto
qed

from 26 have norm(f(k))  $\leq 1 \operatorname{div} \operatorname{abs}(\sin(\pi * k / n))$  by simp
also have  $\dots \leq 1 / \operatorname{abs}(2 * k / n)$ 
proof -
  have  $\sin(\pi * k / n) \geq (2 * k) / n$  using sin-ineq-inst by simp
  moreover have  $(2 * k) / n > 0$  using  $n \langle 1 \leq k \rangle$  by auto
  ultimately have  $\operatorname{abs}(\sin(\pi * k / n)) \geq \operatorname{abs}(2 * k / n)$  by auto
  have  $\operatorname{abs}(2 * k / n) > 0$  using  $\langle 2 * k / n > 0 \rangle$  by linarith
  then show  $1 \operatorname{div} \operatorname{abs}(\sin(\pi * k / n)) \leq 1 / \operatorname{abs}(2 * k / n)$ 
    using  $\langle \operatorname{abs}(2 * k / n) > 0 \rangle \langle \operatorname{abs}(\sin(\pi * k / n)) \geq \operatorname{abs}(2 * k / n) \rangle$ 
    by (intro frac-le) auto
qed
also have  $\dots = n / (2 * k)$  using  $\langle k \geq 1 \rangle$  by simp
finally have norm(f(k))  $\leq n / (2 * k)$  by blast
}
note ineq = this

```

```

have  $\sqrt{n} * \text{norm}(\text{sum } \chi \{1..x\}) < n * \ln n$ 
proof (cases even n)
  case True
  have  $\text{norm}(f(n \text{ div } 2)) \leq 1$ 
  proof -
    have  $\text{int}(n \text{ div } 2) \geq 1$  using  $n \langle \text{even } n \rangle$  by auto
    show ?thesis
      using  $\text{ineq}[OF \langle \text{int}(n \text{ div } 2) \geq 1 \rangle]$  True n by force
  qed
from 24 have  $\sqrt{n} * \text{norm}(\text{sum } \chi \{1..x\})$ 
   $\leq (\sum k = 1..<n. \text{norm}(f(\text{int } k)))$  by blast
also have  $\dots = (\sum k = 1..n-1. \text{norm}(f(\text{int } k)))$ 
  by (intro sum.cong) auto
also have  $\dots \leq 2 * (\sum k = 1..(n-2) \text{ div } 2. \text{norm}(f(\text{int } k))) + \text{norm}(f(n$ 

div 2))
real n div (2*k))

using 25(2)[OF True] by blast
also have  $\dots \leq \text{real } n * (\sum k = 1..(n-2) \text{ div } 2. 1 / k) + \text{norm}(f(n \text{ div } 2))$ 
proof -
  have  $(\sum k = 1..(n-2) \text{ div } 2. \text{norm}(f(\text{int } k))) \leq (\sum k = 1..(n-2) \text{ div } 2.$ 

real n div (2*k))
(real n div 2) * (1 / k))

proof (rule sum-mono)
    fix k
    assume  $k \in \{1..(n-2) \text{ div } 2\}$ 
    then have  $1 \leq \text{int } k$   $\text{int } k \leq n \text{ div } 2$  by auto
    show  $\text{norm}(f(\text{int } k)) \leq \text{real } n / (2*k)$ 
      using  $\text{ineq}[OF \langle 1 \leq \text{int } k \rangle \langle \text{int } k \leq n \text{ div } 2 \rangle]$  by auto
  qed
also have  $\dots = (\sum k = 1..(n-2) \text{ div } 2. (\text{real } n \text{ div } 2) * (1 / k))$ 
  by (rule sum.cong,auto)
also have  $\dots = (\text{real } n \text{ div } 2) * (\sum k = 1..(n-2) \text{ div } 2. 1 / k)$ 
  using sum-distrib-left[symmetric] by fast
finally have  $(\sum k = 1..(n-2) \text{ div } 2. \text{norm}(f(\text{int } k))) \leq$ 

$(\text{real } n \text{ div } 2) * (\sum k = 1..(n-2) \text{ div } 2. 1 / k)$ 
(real n div 2) * (sum k = 1..(n-2) div 2. 1 / k)

by blast
then show ?thesis by argo
qed
also have  $\dots = \text{real } n * \text{harm}((n-2) \text{ div } 2) + \text{norm}(f(n \text{ div } 2))$ 
  unfolding harm-def inverse-eq-divide by simp
also have  $\dots < n * \ln n$ 
proof (cases n = 2)
  case True
  have  $\text{real } n * \text{harm}((n-2) \text{ div } 2) + \text{norm}(f(\text{int}(n \text{ div } 2))) \leq 1$ 
  using  $\langle n = 2 \rangle \langle \text{norm}(f(\text{int}(n \text{ div } 2))) \leq 1 \rangle$ 
  unfolding harm-def by simp
  moreover have  $\text{real } n * \ln(\text{real } n) \geq 4 / 3$ 
  using  $\langle n = 2 \rangle \text{ln2-ge-two-thirds}$  by auto
  ultimately show ?thesis by argo
next

```

```

case False
have  $n > 3$  using  $n \langle n \neq 2 \rangle \langle \text{even } n \rangle$  by auto
then have  $(n-2) \text{ div } 2 > 0$  by simp
then have  $\text{harm } ((n-2) \text{ div } 2) < \ln (\text{real } (2 * ((n-2) \text{ div } 2) + 1))$ 
  using harm-less-ln by blast
also have  $\dots = \ln (\text{real } (n-1))$ 
  using  $\langle \text{even } n \rangle \langle n > 3 \rangle$  by simp
finally have 1:  $\text{harm } ((n-2) \text{ div } 2) < \ln (\text{real } (n-1))$ 
  by blast
then have  $\text{real } n * \text{harm } ((n-2) \text{ div } 2) < \text{real } n * \ln (\text{real } (n-1))$ 
  using  $n$  by simp
then have  $\text{real } n * \text{harm } ((n-2) \text{ div } 2) + \text{norm } (f (\text{int } (n \text{ div } 2)))$ 
   $< \text{real } n * \ln (\text{real } (n-1)) + 1$ 
  using  $\langle \text{norm } (f (\text{int } (n \text{ div } 2))) \leq 1 \rangle$  by argo
also have  $\dots = \text{real } n * \ln (\text{real } (n-1)) + \text{real } n * 1 / \text{real } n$ 
  using  $n$  by auto
also have  $\dots < \text{real } n * \ln (\text{real } (n-1)) + \text{real } n * \ln (1 + 1 / (\text{real } n - 1))$ 
proof -
  have  $\text{real } n > 1$   $\text{real } n > 0$  using  $n$  by simp+
  then have  $\text{real } n * (1 / \text{real } n) < \text{real } n * \ln (1 + 1 / (\text{real } n - 1))$ 
    by (intro mult-strict-left-mono harm-aux-ineq-1) auto
  then show ?thesis by auto
qed
also have  $\dots = \text{real } n * (\ln (\text{real } (n-1)) + \ln (1 + 1 / (\text{real } n - 1)))$ 
  by argo
also have  $\dots = \text{real } n * (\ln (\text{real } (n-1)) * (1 + 1 / (\text{real } n - 1)))$ 
proof -
  have  $\text{real } (n-1) > 0$   $1 + 1 / (\text{real } n - 1) > 0$ 
    using  $n$  by (auto simp add: add-pos-nonneg)
  show ?thesis
    by (subst ln-mult [OF \langle \text{real } (n-1) > 0 \rangle \langle 1 + 1 / (\text{real } n - 1) >
0],symmetric],blast)
qed
also have  $\dots = \text{real } n * \ln n$ 
  using  $n$  by (auto simp add: divide-simps)
finally show ?thesis by blast
qed
finally show ?thesis by blast
next
case False
from 24 have  $\text{sqrt } n * \text{norm } (\text{sum } \chi \{1..x\}) \leq (\sum k=1..<n. \text{norm } (f (\text{int } k)))$ 
  by blast
also have  $\dots = (\sum k=1..n-1. \text{norm } (f (\text{int } k)))$ 
  by (intro sum.cong) auto
also have  $\dots \leq 2 * (\sum k=1..(n-1) \text{ div } 2. \text{norm } (f (\text{int } k)))$ 
  using 25(1)[OF False] by blast
also have  $\dots \leq \text{real } n * (\sum k=1..(n-1) \text{ div } 2. 1 / k)$ 
proof -
  have  $(\sum k=1..(n-1) \text{ div } 2. \text{norm } (f (\text{int } k))) \leq (\sum k=1..(n-1) \text{ div } 2.$ 

```

```

real n div (2*k))
  proof (rule sum-mono)
    fix k
    assume k ∈ {1..(n - 1) div 2}
    then have 1 ≤ int k int k ≤ n div 2 by auto
    show norm (f (int k)) ≤ real n / (2*k)
      using ineq[OF ⟨1 ≤ int k ⟨int k ≤ n div 2⟩] by auto
  qed
  also have ... = (∑ k = 1..(n - 1) div 2. (n / 2) * (1 / k))
    by (rule sum.cong,auto)
  also have ... = (n / 2) * (∑ k = 1..(n - 1) div 2. 1 / k)
    using sum-distrib-left[symmetric] by fast
  finally have (∑ k = 1..(n - 1) div 2. norm (f (int k))) ≤
    (real n div 2) * (∑ k = 1..(n - 1) div 2. 1 / k)
    by blast
  then show ?thesis by argo
qed
also have ... = real n * harm ((n - 1) div 2)
  unfolding harm-def inverse-eq-divide by simp
also have ... < n * ln n
proof -
  have n > 2 using n ⟨odd n⟩ by presburger
  then have (n-1) div 2 > 0 by auto
  then have harm ((n - 1) div 2) < ln (real (2 * ((n - 1) div 2) + 1))
    using harm-less-ln by blast
  also have ... = ln (real n) using ⟨odd n⟩ by simp
  finally show ?thesis using n by simp
qed
finally show ?thesis by blast
qed

then have 1: sqrt n * norm (sum χ {1..x}) < n * ln n
  by blast
show norm (sum χ {1..x}) < sqrt n * ln n
proof -
  have 2: norm (sum χ {1..x}) * sqrt n < n * ln n
    using 1 by argo
  have sqrt n > 0 using n by simp
  have 3: (n * ln n) / sqrt n = sqrt n * ln n
    using n by (simp add: field-simps)
  show norm (sum χ {1..x}) < sqrt n * ln n
    using mult-imp-less-div-pos[OF ⟨sqrt n > 0⟩ 2] 3 by argo
qed
qed

```

8.2 General case

We now first prove the inequality for the general case in terms of the divisor function:

theorem (in *dcharacter*) *polya-vinogradov-inequality-explicit*:
assumes *nonprincipal*: $\chi \neq \text{principal-dchar } n$
shows $\text{norm } (\text{sum } \chi \{1..x\}) < \text{sqrt conductor} * \ln \text{conductor} * \text{divisor-count}$
(*n div conductor*)
proof –
write *primitive-extension* (Φ)
write *conductor* (*c*)
interpret Φ : *primitive-dchar c residue-mult-group c primitive-extension*
using *primitive-primitive-extension nonprincipal* **by** *metis*

have $*$: $k \leq x \text{ div } b \iff b * k \leq x$ **if** $b > 0$ **for** $b \ k$
by (*metis that antisym-conv div-le-mono div-mult-self1-is-m less-imp-le not-less times-div-less-eq-dividend*)

have $**$: $a > 0$ **if** $a \text{ dvd } n$ **for** a
using n **that** **by** (*auto intro!: Nat.gr0I*)

from *nonprincipal* **have** $(\text{sum } m=1..x. \chi \ m) = (\text{sum } m \mid m \in \{1..x\} \wedge \text{coprime } m \ n. \Phi \ m)$
by (*intro sum.mono-neutral-cong-right*) (*auto simp: eq-zero-iff principal-decomposition*)
also **have** $\dots = (\text{sum } m=1..x. \Phi \ m * (\text{sum } d \mid d \text{ dvd gcd } m \ n. \text{moebius-mu } d))$
by (*subst sum-moebius-mu-divisors'*, *intro sum.mono-neutral-cong-left*)
(*auto simp: coprime-iff-gcd-eq-1 simp del: coprime-imp-gcd-eq-1*)
also **have** $\dots = (\text{sum } m=1..x. \text{sum } d \mid d \text{ dvd gcd } m \ n. \Phi \ m * \text{moebius-mu } d)$
by (*simp add: sum-distrib-left*)
also **have** $\dots = (\text{sum } m=1..x. \text{sum } d \mid d \text{ dvd } m \wedge d \text{ dvd } n. \Phi \ m * \text{moebius-mu } d)$
by (*intro sum.cong*) *auto*
also **have** $\dots = (\text{sum } (m, d) \in (\text{SIGMA } m: \{1..x\}. \{d. d \text{ dvd } m \wedge d \text{ dvd } n\}). \Phi \ m * \text{moebius-mu } d)$
using n **by** (*subst sum.Sigma*) *auto*
also **have** $\dots = (\text{sum } (d, q) \in (\text{SIGMA } d: \{d. d \text{ dvd } n\}. \{1..x \text{ div } d\}). \text{moebius-mu } d * \Phi \ (d * q))$
by (*intro sum.reindex-bij-witness*[*of* - $\lambda(d, q). (d * q, d) \lambda(m, d). (d, m \text{ div } d)$])
(*auto simp: * ** Suc-le-eq*)
also **have** $\dots = (\text{sum } d \mid d \text{ dvd } n. \text{moebius-mu } d * \Phi \ d * (\text{sum } q=1..x \text{ div } d. \Phi \ q))$
using n **by** (*subst sum.Sigma* [*symmetric*]) (*auto simp: sum-distrib-left mult.assoc*)
finally **have** *eq*: $(\text{sum } m=1..x. \chi \ m) = \dots$.

have $\text{norm } (\text{sum } m=1..x. \chi \ m) \leq$
 $(\text{sum } d \mid d \text{ dvd } n. \text{norm } (\text{moebius-mu } d * \Phi \ d) * \text{norm } (\text{sum } q=1..x \text{ div } d. \Phi \ q))$
unfolding *eq* **by** (*intro sum-norm-le*) (*simp add: norm-mult*)
also **have** $\dots < (\text{sum } d \mid d \text{ dvd } n. \text{norm } (\text{moebius-mu } d * \Phi \ d) * (\text{sqrt } c * \ln \ c))$
(*is sum ?lhs - < sum ?rhs -*)
proof (*rule sum-strict-mono-ex1*)
show $\forall d \in \{d. d \text{ dvd } n\}. ?lhs \ d \leq ?rhs \ d$
by (*intro ballI mult-left-mono less-imp-le*[*OF* Φ .*polya-vinogradov-inequality-primitive*])
auto
show $\exists d \in \{d. d \text{ dvd } n\}. ?lhs \ d < ?rhs \ d$
by (*intro ballI*[*of* - *1*] *mult-strict-left-mono* Φ .*polya-vinogradov-inequality-primitive*)
auto

```

qed (use n in auto)
also have ... = sqrt c * ln c * (∑ d | d dvd n. norm (moebius-mu d * Φ d))
  by (simp add: sum-distrib-left sum-distrib-right mult-ac)
also have (∑ d | d dvd n. norm (moebius-mu d * Φ d)) =
  (∑ d | d dvd n ∧ squarefree d ∧ coprime d c. 1)
  using n by (intro sum.mono-neutral-cong-right)
  (auto simp: moebius-mu-def Φ.eq-zero-iff norm-mult norm-power
Φ.norm)
also have ... = card {d. d dvd n ∧ squarefree d ∧ coprime d c}
  by simp
also have card {d. d dvd n ∧ squarefree d ∧ coprime d c} ≤ card {d. d dvd (n
div c)}
proof (intro card-mono; safe?)
  show finite {d. d dvd (n div c)}
    using dvd-div-eq-0-iff[of c n] n conductor-dvd by (intro finite-divisors-nat)
auto
next
fix d assume d: d dvd n squarefree d coprime d c
hence d > 0 by (intro Nat.gr0I) auto
show d dvd (n div c)
proof (rule multiplicity-le-imp-dvd)
  fix p :: nat assume p: prime p
  show multiplicity p d ≤ multiplicity p (n div c)
  proof (cases p dvd d)
    assume p dvd d
    with d ⟨d > 0⟩ p have multiplicity p d = 1
      by (auto simp: squarefree-factorial-semiring' in-prime-factors-iff)
    moreover have p dvd (n div c)
  proof –
    have p dvd c * (n div c)
      using ⟨p dvd d⟩ ⟨d dvd n⟩ conductor-dvd by auto
    moreover have ¬(p dvd c)
      using d p ⟨p dvd d⟩ coprime-common-divisor not-prime-unit by blast
    ultimately show p dvd (n div c)
      using p prime-dvd-mult-iff by blast
  qed
hence multiplicity p (n div c) ≥ 1
  using n p conductor-dvd dvd-div-eq-0-iff[of c n]
  by (intro multiplicity-geI) (auto intro: Nat.gr0I)
  ultimately show ?thesis by simp
qed (auto simp: not-dvd-imp-multiplicity-0)
qed (use ⟨d > 0⟩ in simp-all)
qed
also have card {d. d dvd (n div c)} = divisor-count (n div c)
  by (simp add: divisor-count-def)
finally show norm (sum χ {1..x}) < sqrt c * ln c * divisor-count (n div c)
  using conductor-gr-0 by (simp add: mult-left-mono)
qed

```

Next, we obtain a suitable upper bound on the number of divisors of n :

```

lemma divisor-count-upper-bound-aux:
  fixes  $n :: \text{nat}$ 
  shows  $\text{divisor-count } n \leq 2 * \text{card } \{d. d \text{ dvd } n \wedge d \leq \text{sqrt } n\}$ 
proof (cases  $n = 0$ )
  case False
  hence  $n: n > 0$  by simp
  have  $*$ :  $x > 0$  if  $x \text{ dvd } n$  for  $x$ 
    using that  $n$  by (auto intro!: Nat.gr0I)
  have  $**$ :  $\text{real } n = \text{sqrt } (\text{real } n) * \text{sqrt } (\text{real } n)$ 
    by simp
  have  $***$ :  $n < x * \text{sqrt } n \iff \text{sqrt } n < x * \text{sqrt } n < n \iff x < \text{sqrt } n$  for  $x$ 
    by (metis  $**$  n of-nat-0-less-iff mult-less-iff1 real-sqrt-gt-0-iff) $+$ 

  have  $\text{divisor-count } n = \text{card } \{d. d \text{ dvd } n\}$ 
    by (simp add: divisor-count-def)
  also have  $\{d. d \text{ dvd } n\} = \{d. d \text{ dvd } n \wedge d \leq \text{sqrt } n\} \cup \{d. d \text{ dvd } n \wedge d > \text{sqrt } n\}$ 
    by auto
  also have  $\text{card } \dots = \text{card } \{d. d \text{ dvd } n \wedge d \leq \text{sqrt } n\} + \text{card } \{d. d \text{ dvd } n \wedge d > \text{sqrt } n\}$ 
    using  $n$  by (subst card-Un-disjoint) auto
  also have bij-betw  $(\lambda d. n \text{ div } d)$   $\{d. d \text{ dvd } n \wedge d > \text{sqrt } n\}$   $\{d. d \text{ dvd } n \wedge d < \text{sqrt } n\}$ 
    using  $n$  by (intro bij-betwI[of - -  $\lambda d. n \text{ div } d$ ])
      (auto simp: Real.real-of-nat-div real-sqrt-divide field-simps * ***)
  hence  $\text{card } \{d. d \text{ dvd } n \wedge d > \text{sqrt } n\} = \text{card } \{d. d \text{ dvd } n \wedge d < \text{sqrt } n\}$ 
    by (rule bij-betw-same-card)
  also have  $\dots \leq \text{card } \{d. d \text{ dvd } n \wedge d \leq \text{sqrt } n\}$ 
    using  $n$  by (intro card-mono) auto
  finally show  $\text{divisor-count } n \leq 2 * \dots$  by simp
qed auto

lemma divisor-count-upper-bound:
  fixes  $n :: \text{nat}$ 
  shows  $\text{divisor-count } n \leq 2 * \text{nat } \lfloor \text{sqrt } n \rfloor$ 
proof (cases  $n = 0$ )
  case False
  have  $\text{divisor-count } n \leq 2 * \text{card } \{d. d \text{ dvd } n \wedge d \leq \text{sqrt } n\}$ 
    by (rule divisor-count-upper-bound-aux)
  also have  $\text{card } \{d. d \text{ dvd } n \wedge d \leq \text{sqrt } n\} \leq \text{card } \{1.. \text{nat } \lfloor \text{sqrt } n \rfloor\}$ 
    using False by (intro card-mono) (auto simp: le-nat-iff le-floor-iff Suc-le-eq intro!: Nat.gr0I)
  also have  $\dots = \text{nat } \lfloor \text{sqrt } n \rfloor$  by simp
  finally show ?thesis by simp
qed auto

lemma divisor-count-upper-bound':
  fixes  $n :: \text{nat}$ 
  shows  $\text{real } (\text{divisor-count } n) \leq 2 * \text{sqrt } n$ 

```


proof –
have $\text{real}(\text{divisor-count } n) \leq 2 * \text{real}(\text{nat } \lfloor \text{sqrt } n \rfloor)$
using *divisor-count-upper-bound*[of n] **by** *linarith*
also have $\dots \leq 2 * \text{sqrt } n$
by *simp*
finally show *?thesis* .
qed

We are now ready to prove the ‘regular’ Pólya–Vinogradov inequality. Apostol formulates it in the following way (Theorem 13.15, notation adapted): ‘If χ is any nonprincipal character mod n , then for all $x \geq 2$ we have $\sum_{m \leq x} \chi(m) = O(\sqrt{n} \log n)$.’

The precondition $x \geq 2$ here is completely unnecessary. The ‘Big-O’ notation is somewhat problematic since it does not make explicit in what way the variables are quantified (in particular the x and the χ). The statement of the theorem in this way (for a fixed character χ) seems to suggest that n is fixed here, which would make the use of ‘Big-O’ completely vacuous, since it is an asymptotic statement about n .

We therefore decided to formulate the inequality in the following more explicit way, even giving an explicit constant factor:

theorem (in *dcharacter*) *polya-vinogradov-inequality*:
assumes *nonprincipal*: $\chi \neq \text{principal-dchar } n$
shows $\text{norm}(\sum_{m=1..x} \chi m) < 2 * \text{sqrt } n * \ln n$
proof –
have $n \text{ div conductor} > 0$
using *n conductor-dvd dvd-div-eq-0-iff*[of *conductor n*] **by** *auto*
have $\text{norm}(\sum_{m=1..x} \chi m) < \text{sqrt conductor} * \ln \text{conductor} * \text{divisor-count}(n \text{ div conductor})$
using *nonprincipal* **by** (*rule polya-vinogradov-inequality-explicit*)
also have $\dots \leq \text{sqrt conductor} * \ln \text{conductor} * (2 * \text{sqrt}(n \text{ div conductor}))$
using *conductor-gr-0* ($n \text{ div conductor} > 0$)
by (*intro mult-left-mono divisor-count-upper-bound'*) (*auto simp: Suc-le-eq*)
also have $\text{sqrt}(n \text{ div conductor}) = \text{sqrt } n / \text{sqrt conductor}$
using *conductor-dvd* **by** (*simp add: Real.real-of-nat-div real-sqrt-divide*)
also have $\text{sqrt conductor} * \ln \text{conductor} * (2 * (\text{sqrt } n / \text{sqrt conductor})) = 2 * \text{sqrt } n * \ln \text{conductor}$
using *conductor-gr-0 n* **by** (*simp add: algebra-simps*)
also have $\dots \leq 2 * \text{sqrt } n * \ln n$
using *conductor-le-modulus conductor-gr-0* **by** (*intro mult-left-mono*) *auto*
finally show *?thesis* .
qed

unbundle *vec-lambda-notation*

end

References

- [1] T. M. Apostol. *Introduction to Analytic Number Theory*. Undergraduate Texts in Mathematics. Springer-Verlag, 1976.