Dirichlet L-functions and Dirichlet's Theorem

Manuel Eberl

March 17, 2025

Abstract

This article provides a formalisation of Dirichlet characters and Dirichlet *L*-functions including proofs of their basic properties – most notably their analyticity, their areas of convergence, and their nonvanishing for $\Re(s) \geq 1$. All of this is built in a very high-level style using Dirichlet series. The proof of the non-vanishing follows a very short and elegant proof by Newman [4], which we attempt to reproduce faithfully in a similar level of abstraction in Isabelle.

This also leads to a relatively short proof of Dirichlet's Theorem, which states that, if h and n are coprime, there are infinitely many primes p with $p \equiv h \pmod{n}$.

Contents

1	Mu	Itiplicative Characters of Finite Abelian Groups	3
	1.1	Definition of characters	3
	1.2	Basic properties	4
	1.3	The Character group	6
	1.4	The isomorphism between a group and its dual	9
	1.5	Non-trivial facts about characters	24
	1.6	The first orthogonality relation	35
	1.7	The isomorphism between a group and its double dual $\ . \ . \ .$	36
2	Dirichlet Characters		39
	2.1	The multiplicative group of residues	39
	2.2	Definition of Dirichlet characters	41
	2.3	Sums of Dirichlet characters	46
3	Dirichlet L-functions		49
	3.1	Definition and basic properties	49
	3.2	The non-vanishing for $\Re(s) \ge 1$	59
	3.3	Asymptotic bounds on partial sums of Dirichlet L functions .	69
	3.4	Evaluation of $L(\chi, 0)$	72
	3.5	Properties of $L(\chi, s)$ for real χ	73
4	Dirichlet's Theorem on primes in arithmetic progressions		75
	4.1	Auxiliary results	75
	4.2	The contribution of the non-principal characters	76
	4.3	The contribution of the principal character	78
	4.4	The main result	80

1 Multiplicative Characters of Finite Abelian Groups

theory Multiplicative-Characters imports Complex-Main Finitely-Generated-Abelian-Groups.Finitely-Generated-Abelian-Groups begin

notation integer-mod-group $(\langle Z \rangle)$

1.1 Definition of characters

A (multiplicative) character is a completely multiplicative function from a group to the complex numbers. For simplicity, we restrict this to finite abelian groups here, which is the most interesting case.

Characters form a group where the identity is the *principal* character that maps all elements to 1, multiplication is point-wise multiplication of the characters, and the inverse is the point-wise complex conjugate.

This group is often called the *Pontryagin dual* group and is isomorphic to the original group (in a non-natural way) while the double-dual group *is* naturally isomorphic to the original group.

To get extensionality of the characters, we also require characters to map anything that is not in the group to 0.

definition principal-char :: ('a, 'b) monoid-scheme $\Rightarrow 'a \Rightarrow$ complex where principal-char $G \ a = (if \ a \in carrier \ G \ then \ 1 \ else \ 0)$

definition *inv-character* where *inv-character* $\chi = (\lambda a. cnj (\chi a))$

lemma inv-character-principal [simp]: inv-character (principal-char G) = principal-char G

 $\mathbf{by} \ (simp \ add: \ inv-character-def \ principal-char-def \ fun-eq-iff)$

lemma inv-character-inv-character [simp]: inv-character (inv-character χ) = χ by (simp add: inv-character-def)

lemma eval-inv-character: inv-character $\chi j = cnj (\chi j)$ by (simp add: inv-character-def)

bundle character-syntax begin notation principal-char $(\langle \chi_{0} \rangle)$ end

locale character = finite-comm-group + fixes $\chi :: 'a \Rightarrow complex$ assumes char-one-nz: $\chi \mathbf{1} \neq 0$ assumes char-eq-0: $a \notin carrier \ G \Longrightarrow \chi \ a = 0$ assumes char-mult [simp]: $a \in carrier \ G \Longrightarrow b \in carrier \ G \Longrightarrow \chi \ (a \otimes b) = \chi$ $a * \chi b$ begin

1.2 Basic properties

```
lemma char-one [simp]: \chi \mathbf{1} = 1
proof-
 from char-mult[of 1 1] have \chi \mathbf{1} * (\chi \mathbf{1} - 1) = 0
   by (auto simp del: char-mult)
 with char-one-nz show ?thesis by simp
qed
lemma char-power [simp]: a \in carrier \ G \Longrightarrow \chi \ (a \ [] k) = \chi \ a \ k
 by (induction k) auto
lemma char-root:
 assumes a \in carrier G
 shows \chi a \cap ord a = 1
proof –
  from assess have \chi a \widehat{} ord a = \chi (a [\widehat{}] ord a)
   by (subst char-power) auto
 also from fin and assms have a [ ] ord a = 1 by (intro pow-ord-eq-1) auto
 finally show ?thesis by simp
qed
lemma char-root':
 assumes a \in carrier G
 shows \chi a \cap order G = 1
proof -
 from assms have \chi a \widehat{} order G = \chi (a [\widehat{}] order G) by simp
 also from fin and assms have a [\uparrow] order G = 1 by (intro pow-order-eq-1) auto
 finally show ?thesis by simp
qed
lemma norm-char: norm (\chi \ a) = (if \ a \in carrier \ G \ then \ 1 \ else \ 0)
proof (cases a \in carrier G)
 case True
 have norm (\chi \ a) \widehat{} order G = norm (\chi \ a \widehat{} order G) by (simp add: norm-power)
 also from True have \chi a \widehat{} order G = 1 by (rule char-root')
 finally have norm (\chi \ a) \widehat{} order G = 1 \widehat{} order G by simp
 hence norm (\chi a) = 1 by (subst (asm) power-eq-iff-eq-base) auto
  with True show ?thesis by auto
\mathbf{next}
  case False
 thus ?thesis by (auto simp: char-eq-0)
qed
```

lemma char-eq-0-iff: $\chi \ a = 0 \iff a \notin carrier G$ **proof** – **have** $\chi \ a = 0 \iff norm \ (\chi \ a) = 0$ **by** simp **also have** ... $\iff a \notin carrier G$ **by** (subst norm-char) auto **finally show** ?thesis . **qed**

```
lemma inv-character: character G (inv-character \chi)
by standard (auto simp: inv-character-def char-eq-0)
```

lemma mult-inv-character: $\chi \ k * inv$ -character $\chi \ k = principal$ -char $G \ k$ **proof** – **have** $\chi \ k * inv$ -character $\chi \ k = of$ -real (norm ($\chi \ k$) ^ 2) **by** (subst complex-norm-square) (simp add: inv-character-def) **also have** ... = principal-char $G \ k$ **by** (simp add: principal-char-def norm-char) **finally show** ?thesis . **qed**

lemma

assumes $a \in carrier \ G$ shows $char-inv: \chi \ (inv \ a) = cnj \ (\chi \ a)$ and $char-inv': \chi \ (inv \ a) = inverse \ (\chi \ a)$ proof – from assms have $inv \ a \otimes a = 1$ by simpalso have $\chi \dots = 1$ by simpalso from assms have $\chi \ (inv \ a \otimes a) = \chi \ (inv \ a) * \chi \ a$ by $(intro \ char-mult)$ auto finally have $*: \chi \ (inv \ a) * \chi \ a = 1$. thus $\chi \ (inv \ a) = inverse \ (\chi \ a)$ by $(auto \ simp: \ divide-simps)$ also from $mult-inv-character[of \ a]$ and assms have $inverse \ (\chi \ a) = cnj \ (\chi \ a)$ by $(auto \ simp \ add: inv-character-def \ principal-char-def \ divide-simps \ mult.commute)$ finally show $\chi \ (inv \ a) = cnj \ (\chi \ a)$.

qou

end

lemma (in finite-comm-group) character-principal [simp, intro]: character G (principal-char G)

by standard (auto simp: principal-char-def)

lemmas [simp, intro] = finite-comm-group.character-principal

lemma character-ext: **assumes** character $G \ \chi$ character $G \ \chi' \ \Lambda x. \ x \in carrier \ G \Longrightarrow \chi \ x = \chi' \ x$ **shows** $\chi = \chi'$ **proof fix** x :: 'a show $\chi x = \chi' x$ using assms by (cases $x \in carrier G$) (auto simp: character.char-eq- θ) qed lemma character-mult [intro]: assumes character $G \chi$ character $G \chi'$ shows character $G (\lambda x. \chi x * \chi' x)$ proof – interpret χ : character $G \chi$ by fact interpret χ : character $G \chi'$ by fact show ?thesis by standard (auto simp: χ .char-eq- θ) qed

lemma character-inv-character-iff [simp]: character G (inv-character χ) \longleftrightarrow character G χ **proof assume** character G (inv-character χ) **from** character.inv-character [OF this] **show** character G χ **by** simp **qed** (auto simp: character.inv-character)

definition characters :: ('a, 'b) monoid-scheme \Rightarrow ('a \Rightarrow complex) set where characters $G = \{\chi. \text{ character } G \ \chi\}$

1.3 The Character group

The characters of a finite abelian group G form another group \widehat{G} , which is called its Pontryagin dual group. This generalises to the more general setting of locally compact abelian groups, but we restrict ourselves to the finite setting because it is much easier.

definition Characters :: ('a, 'b) monoid-scheme \Rightarrow ('a \Rightarrow complex) monoid **where** Characters G = (] carrier = characters G, monoid.mult = $(\lambda \chi_1 \ \chi_2 \ k. \ \chi_1 \ k \ast \chi_2 \ k)$,

one = principal-char G

lemma carrier-Characters: carrier (Characters G) = characters Gby (simp add: Characters-def)

lemma one-Characters: one (Characters G) = principal-char Gby (simp add: Characters-def)

lemma mult-Characters: monoid.mult (Characters G) $\chi_1 \chi_2 = (\lambda a. \chi_1 a * \chi_2 a)$ by (simp add: Characters-def)

context finite-comm-group
begin

sublocale principal: character G principal-char G ...

lemma finite-characters [intro]: finite (characters G) **proof** (rule finite-subset) **show** characters $G \subseteq (\lambda f x. if x \in carrier G then f x else 0)$ ' Pi_E (carrier G) (λ -. {z. z \cap order G = 1}) (is - \subseteq ?h '?Chars) proof (intro subsetI, goal-cases) case (1χ) then interpret χ : character $G \chi$ by (simp add: characters-def) have ?h (restrict χ (carrier G)) $\in ?h$ '? Chars **by** (*intro imageI*) (*auto simp*: χ .char-root') also have ?h (restrict χ (carrier G)) = χ by (simp add: fun-eq-iff χ .char-eq- θ) finally show ?case . qed **show** finite (?h '?Chars) by (intro finite-imageI finite-PiE finite-roots-unity) (auto simp: Suc-le-eq) qed lemma finite-comm-group-Characters [intro]: finite-comm-group (Characters G) proof fix $\chi \chi'$ assume $*: \chi \in carrier$ (Characters G) $\chi' \in carrier$ (Characters G) **from** * **interpret** χ : character $G \chi$ by (simp-all add: characters-def carrier-Characters) from * interpret χ' : character $G \chi'$ by (simp-all add: characters-def carrier-Characters) have character G (λk . $\chi k * \chi' k$) by standard (insert *, simp-all add: χ .char-eq-0 one-Characters mult-Characters characters-def carrier-Characters) **thus** $\chi \otimes_{Characters G} \chi' \in carrier (Characters G)$ $\mathbf{by} \ (simp \ add: \ characters \ def \ one-Characters \ mult-Characters \ carrier-Characters)$ \mathbf{next} have character G (principal-char G)... thus $\mathbf{1}_{Characters \ G} \in carrier \ (Characters \ G)$ by (simp add: characters-def one-Characters mult-Characters carrier-Characters) next fix χ assume $*: \chi \in carrier$ (Characters G) **from** * **interpret** χ : character $G \chi$ by (simp-all add: characters-def carrier-Characters) show $\mathbf{1}_{Characters \ G} \otimes_{Characters \ G} \chi = \chi$ and $\chi \otimes_{Characters \ G} \mathbf{1}_{Characters \ G}$ $= \chi$ by (simp-all add: principal-char-def fun-eq-iff χ .char-eq-0 one-Characters mult-Characters) next have $\chi \in Units$ (Characters G) if $\chi \in carrier$ (Characters G) for χ proof – from that interpret χ : character $G \chi$ by (simp add: characters-def car*rier-Characters*) have $\chi \otimes_{Characters G}$ inv-character $\chi = \mathbf{1}_{Characters G}$ and inv-character $\chi \otimes_{Characters G} \chi = \mathbf{1}_{Characters G}$ by (simp-all add: χ .mult-inv-character mult-ac one-Characters mult-Characters) **moreover from** that have inv-character $\chi \in carrier$ (Characters G) **by** (*simp add: characters-def carrier-Characters*)

ultimately show ?thesis using that unfolding Units-def by blast qed thus carrier (Characters G) \subseteq Units (Characters G) .. qed (auto simp: principal-char-def one-Characters mult-Characters carrier-Characters)

end

lemma (in character) character-in-order-1: **assumes** order G = 1shows $\chi = principal-char G$ proof from assms have card (carrier $G - \{1\}$) = 0 **by** (*subst card-Diff-subset*) (*auto simp: order-def*) **hence** carrier $G - \{1\} = \{\}$ by (subst (asm) card-0-eq) auto hence carrier $G = \{1\}$ by auto thus ?thesis by (intro ext) (simp-all add: principal-char-def char-eq-0) \mathbf{qed} **lemma** (in *finite-comm-group*) characters-in-order-1: **assumes** order G = 1**shows** characters $G = \{ principal-char G \}$ using character.character-in-order-1 [OF - assms] by (auto simp: characters-def)

 $\begin{array}{l} \textbf{lemma (in character) inv-Characters: inv_{Characters \ G} \ \chi = inv-character \ \chi \\ \textbf{proof} \ - \\ \textbf{interpret } Characters: finite-comm-group \ Characters \ G \ .. \\ \textbf{have } character \ G \ \chi \ .. \\ \textbf{thus } ?thesis \\ \textbf{by (intro \ Characters.inv-equality)} \\ (auto \ simp: \ characters.def \ mult-inv-character \ mult-ac \\ carrier-Characters \ one-Characters \ mult-Characters) \end{array}$

 \mathbf{qed}

lemma (in finite-comm-group) inv-Characters': $\chi \in characters \ G \Longrightarrow inv_{Characters} \ G \ \chi = inv-character \ \chi$ by (intro character.inv-Characters) (auto simp: characters-def)

lemmas (in finite-comm-group) Characters-simps = carrier-Characters mult-Characters one-Characters inv-Characters'

lemma inv-Characters': $\chi \in characters \ G \implies inv_{Characters \ G} \ \chi = inv-character \chi$

using character.inv-Characters[of $G \chi$] by (simp add: characters-def)

1.4 The isomorphism between a group and its dual

We start this section by inspecting the special case of a cyclic group. Here, any character is fixed by the value it assigns to the generating element of the cyclic group. This can then be used to construct a bijection between the nth unit roots and the elements of the character group - implying the other results.

lemma (in *finite-cyclic-group*) **defines** ic: induce-char $\equiv (\lambda c::complex. (\lambda a. if a \in carrier G then c powi get-exp$ gen a else 0) **shows** order-Characters: order (Characters G) = order Gand gen-fixes-char: [character G a; character G b; a gen = b gen] $\implies a = b$ and unity-root-induce-char: $z \cap order \ G = 1 \implies character \ G \ (induce-char \ z)$ proof interpret C: finite-comm-group Characters G using finite-comm-group-Characters define n where n = order Ghence n: n > 0 using order-qt-0 by presburger from *n*-def have nog: n = ord gen using ord-gen-is-group-order by simp have $xnz: x \neq 0$ if $x \cap n = 1$ for x::complex using n(1) that by (metis *zero-neq-one zero-power*) have m: x powi m = x powi $(m \mod n)$ if $x \cap n = 1$ for x::complex and m::int using powi-mod[OF that n]. **show** cf: character G (induce-char x) if x: $x \cap n = 1$ for x proof show induce-char $x \mathbf{1} \neq 0$ using xnz[OF that] unfolding ic by auto show induce-char x = 0 if $a \notin carrier G$ for a using that unfolding ic by simp **show** induce-char $x (a \otimes b) = induce$ -char x a * induce-char x bif $a \in carrier \ G \ b \in carrier \ G$ for $a \ b$ proof – have x powi get-exp gen $(a \otimes b) = x$ powi get-exp gen a * x powi get-exp gen bproof – have x powi get-exp gen $(a \otimes b) = x$ powi ((get-exp gen a + get-exp gen b) mod n) using m[OF x] get-exp-mult-mod[OF that] n-def ord-gen-is-group-order by metis also have $\ldots = x powi (get-exp gen a + get-exp gen b)$ using m[OF x] by presburger finally show ?thesis by (simp add: power-int-add xnz[OF x]) aed thus ?thesis using that unfolding ic by simp qed qed **define** get-c where gc: get-c = (λc ::'a \Rightarrow complex. c gen) have biji: bij-betw induce-char $\{z, z \cap n = 1\}$ (characters G) and bijg: bij-betw get-c (characters G) $\{z, z \cap n = 1\}$

proof (*intro bij-betwI*[*of* - - - *get-c*]) show iin: induce-char $\in \{z, z \cap n = 1\} \rightarrow$ characters G using cf unfolding characters-defby blast show gi: get-c (induce-char x) = x if $x \in \{z, z \cap n = 1\}$ for x **proof** (cases n = 1) case True with that have x = 1 by force thus ?thesis unfolding ic gc by simp \mathbf{next} case False have $x: x \cap n = 1$ using that by blast have x powi get-exp gen gen = xproof have x powi get-exp gen gen = x powi (get-exp gen gen mod n) using m[OF]x] **by** blast moreover have $(qet-exp \ qen \ mod \ n) = 1$ proof – have $1 = 1 \mod int \ n$ using False n by auto also have $\ldots = get - exp \ gen \ mod \ n$ by (unfold nog, intro pow-eq-int-mod[OF gen-closed], use get-exp-fulfills[OF gen-closed] in auto) finally show ?thesis by argo \mathbf{qed} **ultimately show** x powi get-exp gen gen = x by simp qed thus ?thesis unfolding ic gc by simp ged **show** gin: get- $c \in$ characters $G \rightarrow \{z, z \cap n = 1\}$ proof have False if get-c c $\hat{n} \neq 1$ character G c for c proof – interpret character G c by fact show False using that(1)[unfolded gc] by (simp add: char-root' n-def) qed thus ?thesis unfolding characters-def by blast qed **show** ig: induce-char (get-c y) = y if y: $y \in$ characters G for y **proof** (cases n = 1) case True hence y = principal-char G using y n-def character.character-in-order-1 characters-def by *auto* thus ?thesis unfolding ic gc principal-char-def by force next case False have $yc: y \in carrier$ (Characters G) using y[unfolded carrier-Characters[symmetric]]

interpret character G y using that unfolding characters-def by simp

have ygo: y gen n = 1 using char-root'[OF gen-closed] n-def by blast have y gen powi get-exp gen a = y a if $a \in carrier G$ for a using that proof(induction rule: generator-induct1) case gen have y gen powi get-exp gen gen = y gen powi (get-exp gen gen mod n) using m[OF ygo] by blast also have $\ldots = y$ gen powi ((1::int) mod n) using get-exp-self[OF gen-closed] nog by argo also have $\ldots = y$ gen powi 1 using False n by simp finally have yg: y gen powi get-exp gen gen = y gen by simpthus ?case . case (step x) have y gen powi get-exp gen $(x \otimes gen) = y$ gen powi (get-exp gen $(x \otimes gen)$) mod n) using m[OF ygo] by blast also have $\dots = y$ gen powi ((get-exp gen x + get-exp gen gen) mod n) **using** get-exp-mult-mod[OF step(1) gen-closed, unfolded nog[symmetric]] by argo also have $\dots = y$ gen powi (get-exp gen x + get-exp gen gen) using m[OF]ygo] **by** presburger also have $\ldots = y$ gen powi get-exp gen x * y gen powi get-exp gen gen **by** (*simp add: char-eq-0-iff power-int-add*) also have $\ldots = y \ x * y \ gen \ using \ yg \ step(2)$ by argo also have $\ldots = y \ (x \otimes gen)$ using step(1) by simpfinally show ?case . qed thus induce-char (get-c y) = y unfolding ic gc using char-eq-0 by auto qed **show** bij-betw get-c (characters G) $\{z, z \cap n = 1\}$ using ig gi iin gin **by** (*auto intro: bij-betwI*) qed with card-roots-unity-eq[OF n] n-def show order (Characters G) = order G unfolding order-def **by** (*metis bij-betw-same-card carrier-Characters*) assume assm: character G a character G b a gen = b gen with $bijq[unfolded\ qc\ characters-def\ bij-betw-def\ inj-on-def]$ show a = b by auto qed Moreover, we can show that a character that assigns a "true" root of unity to the generating element of the group, generates the character group. **lemma** (in *finite-cyclic-group*) *finite-cyclic-group-Characters*:

obtains χ where finite-cyclic-group (Characters G) χ proof – interpret C: finite-comm-group Characters G by (rule finite-comm-group-Characters) define n where n: n = order Ghence $nnz: n \neq 0$ by blast from n have nog: n = ord gen using ord-gen-is-group-order by simp obtain x::complex where $x: x \cap n = 1 \ Mm. [[0 < m; m < n]] \implies x \cap m \neq 1$ using true-nth-unity-root by blast

have xnz: $x \neq 0$ using x n by (metis order-gt-0 zero-neq-one zero-power) have m: x powi m = x powi (m mod n) for m::intusing powi-mod[OF x(1)] nnz by blast let $?f = (\lambda a. if a \in carrier G then x powi (get-exp gen a) else 0)$ have cf: character G ?f using unity-root-induce-char[OF x(1)[unfolded n]]. have fpow: $(?f []_{Characters G} m) a = x powi ((get-exp gen a) * m)$ if $a \in carrier \ G$ for a::'a and m::natusing that **proof**(*unfold Characters-def principal-char-def*, *induction m*) case s: $(Suc \ m)$ have x powi (get-exp gen a * int m) * x powi get-exp gen a = x powi (get-exp gen a * (1 + int m))proof - $\mathbf{fix} \ ma :: \ nat$ have x powi ((1 + int ma) * get-exp gen a) $= x \text{ powi} (\text{qet-exp qen } a + \text{int ma} * \text{qet-exp qen } a) \land 0 \neq x$ **by** (*simp add: comm-semirinq-class.distrib xnz*) then show x powi (get-exp gen a * int ma) * x powi get-exp gen a = x powi (get-exp gen a * (1 + int ma))by (simp add: mult.commute power-int-add) ged thus ?case using s by simp qed simp interpret cyclic-group Characters G ?f **proof** (*intro C.element-ord-generates-cyclic*) **show** fc: ?f \in carrier (Characters G) using cf carrier-Characters[of G] characters-def by fast from x nnz have fno: ?f [^]_{Characters G} $m \neq \mathbf{1}_{Characters G}$ if 0 < m m < nfor m**proof** (cases n = 1) case False have $\mathbf{1}_{Characters \ G} \ gen = 1$ unfolding Characters-def principal-char-def using that by simp moreover have $(?f[]_{Characters G} m) gen \neq 1$ proof have $(?f []_{Characters G} m) gen = x powi ((get-exp gen gen) * m)$ using fpow by blast also have $\dots = (x \text{ powi } (\text{get-exp gen gen})) \ \widehat{} m \mathbf{by} (\text{simp add: power-int-mult})$ also have $\ldots = x \cap m$ proof – have x powi (get-exp gen gen) = x powi ((get-exp gen gen) mod n) using m by blast **moreover have** $((get-exp \ gen \ gen) \ mod \ n) = 1$ proof have $1 = 1 \mod int \ n$ using False nnz by simp also have $\ldots = get\text{-}exp \ gen \ mod \ n$ **by** (unfold nog. intro pow-eq-int-mod[OF gen-closed], use get-exp-fulfills[OF gen-closed] in auto) finally show ?thesis by argo

```
qed
       ultimately have x powi (get-exp gen gen) = x by simp
       thus ?thesis by simp
      qed
      finally show ?thesis using x(2)[OF that] by argo
    qed
    ultimately show ?thesis by fastforce
   qed (use that in blast)
   have C.ord ?f = n
   proof -
    from nnz have C.ord ?f \leq n unfolding n
       using C.ord-dvd-group-order[OF fc] order-Characters dvd-nat-bounds by
auto
    with C.ord-conv-Least[OF fc] C.pow-order-eq-1[OF fc] n nnz show C.ord ?f
= n
    by (metis (no-types, lifting) C.ord-pos C.pow-ord-eq-1 fc fno le-neq-implies-less)
   qed
   thus C.ord ? f = order (Characters G) using n order-Characters by argo
 qed
 have finite-cyclic-group (Characters G) ? f by unfold-locales
 with that show ?thesis by blast
qed
```

And as two cyclic groups of the same order are isomorphic it follows the isomorphism of a finite cyclic group and its dual.

 $\begin{array}{l} \textbf{lemma (in finite-cyclic-group) Characters-iso:} \\ G \cong Characters \ G \\ \textbf{proof} \ - \\ \textbf{from finite-cyclic-group-Characters obtain f where f: finite-cyclic-group (Characters G) f . \\ \textbf{then interpret C: finite-cyclic-group Characters G f . \\ \textbf{have cyclic-group (Characters G) f by unfold-locales \\ \textbf{from iso-cyclic-groups-same-order[OF this order-Characters[symmetric]] show } \\ ?thesis . \\ \textbf{qed} \end{array}$

The character groups of two isomorphic groups are also isomorphic.

lemma (in finite-comm-group) iso-imp-iso-chars: **assumes** $G \cong H$ group H **shows** Characters $G \cong$ Characters H **proof** – **interpret** H: finite-comm-group H by (rule iso-imp-finite-comm[OF assms]) from assms have $H \cong G$ using iso-sym by auto then obtain g where g: $g \in iso$ H G unfolding is-iso-def by blast then interpret ggh: group-hom H G g by (unfold-locales, unfold iso-def, simp) **let** ? $f = (\lambda c \ a. \ if \ a \in carrier \ H \ then \ (c \circ g) \ a \ else \ 0)$ have ? $f \in iso$ (Characters G) (Characters H) **proof** (intro isoI) **interpret** CG: finite-comm-group Characters G by (intro finite-comm-group-Characters) interpret CH: finite-comm-group Characters H by (intro H. finite-comm-group-Characters) have f-in: ?f $x \in carrier$ (Characters H) if $x \in carrier$ (Characters G) for x **proof** (unfold carrier-Characters characters-def, rule, unfold-locales)

interpret character G x using that characters-def carrier-Characters by blast show (if $\mathbf{1}_H \in carrier \ H \ then \ (x \circ g) \ \mathbf{1}_H \ else \ \theta) \neq \theta$ using g iso-iff by auto show $\bigwedge a. a \notin carrier H \Longrightarrow (if a \in carrier H then (x \circ g) a else 0) = 0$ by simp

show $?f x (a \otimes_H b) = ?f x a * ?f x b$ if $a \in carrier H b \in carrier H$ for a b

using that by auto

qed

x

show $?f \in hom$ (Characters G) (Characters H) **proof** (*intro homI*) **show** ?f $x \in carrier$ (Characters H) **if** $x \in carrier$ (Characters G) **for** xusing f-in[OF that]. **show** ?f $(x \otimes_{Characters} G y) = ?f x \otimes_{Characters} H$?f y if $x \in carrier$ (Characters G) $y \in carrier$ (Characters G) for x yproof interpret x: character G x using that characters-def carrier-Characters by blastinterpret y: character G y using that characters-def carrier-Characters by blastshow ?thesis using that mult-Characters[of G] mult-Characters[of H] by autoqed qed **show** bij-betw ?f (carrier (Characters G)) (carrier (Characters H)) **proof**(*intro bij-betwI*) define f where f = inv-into (carrier H) g hence $f: f \in iso \ G \ H \ using \ H.iso-set-sym[OF \ g]$ by simpthen interpret fgh: group-hom G H f by (unfold-locales, unfold iso-def, simp) let $?g = (\lambda c \ a. \ if \ a \in carrier \ G \ then \ (c \circ f) \ a \ else \ 0)$ show $?f \in carrier$ (Characters G) $\rightarrow carrier$ (Characters H) using f-in by fast **show** $?g \in carrier$ (Characters H) $\rightarrow carrier$ (Characters G) proof have q-in: $?q \ x \in carrier$ (Characters G) if $x \in carrier$ (Characters H) for **proof** (unfold carrier-Characters characters-def, rule, unfold-locales) interpret character H x using that characters-def carrier-Characters by blastshow (if $\mathbf{1}_G \in carrier \ G \ then \ (x \circ f) \ \mathbf{1}_G \ else \ \theta) \neq \theta$ using f iso-iff by auto**show** $\bigwedge a. a \notin carrier G \Longrightarrow (if a \in carrier G then <math>(x \circ f) a else 0) = 0$ by simp **show** $?g x (a \otimes_G b) = ?g x a * ?g x b if a \in carrier G b \in carrier G for$ $a \ b$ using that by auto qed thus ?thesis by simp

qed

show ?f (?g x) = x if x: $x \in carrier$ (Characters H) for x proof interpret character H x using x characters-def carrier-Characters by blast have ?f(?q x) = x a if $a: a \notin carrier H$ for a using a char-eq-0[OF a]by auto **moreover have** ?f (?q x) a = x a if $a: a \in carrier H$ for a proof from a have inv-into (carrier H) g(g a) = a**by** (*simp add: g ggh.inj-iff-trivial-ker ggh.iso-kernel*) thus ?thesis using a f-def by auto qed ultimately show ?thesis by fast qed **show** ?g(?fx) = x if $x: x \in carrier$ (Characters G) for x proof interpret character G x using x characters-def carrier-Characters by blast have ?g(?fx) = x a if $a: a \notin carrier G$ for a using a char-eq- $\theta[OFa]$ by auto **moreover have** ?g(?fx) = x a if $a: a \in carrier G$ for a using a f-def proof – from a have g (inv-into (carrier H) g a) = a **by** (meson f-inv-into-f g ggh.iso-iff subset-iff) thus ?thesis using a f-def fgh.hom-closed by auto qed ultimately show ?thesis by fast qed ged qed thus ?thesis unfolding is-iso-def by blast qed

The following two lemmas characterize the way a character behaves in a direct group product: a character on the product induces characters on each of the factors. Also, any character on the direct product can be decomposed into a pointwise product of characters on the factors.

lemma *DirProds-subchar*:

assumes finite-comm-group (DirProds Gs I) and x: $x \in carrier$ (Characters (DirProds Gs I)) and i: $i \in I$ and I: finite I defines g: $g \equiv (\lambda c. (\lambda i \in I. (\lambda a. c ((\lambda i \in I. \mathbf{1}_{Gs i})(i:=a)))))$ shows character (Gs i) (g x i) proof – interpret DP: finite-comm-group DirProds Gs I by fact interpret xc: character DirProds Gs I x using x unfolding Characters-def characters-def by auto interpret Gi: finite-comm-group Gs i using i DirProds-finite-comm-group-iff[OF I] DP.finite-comm-group-axioms by blast have allg: $\bigwedge i. i \in I \Longrightarrow group (Gs i)$ using DirProds-group-imp-groups[OF DP.is-group]

show ?thesis **proof**(*unfold-locales*) have $(\lambda i \in I. \mathbf{1}_{Gs \ i}) = (\lambda i \in I. \mathbf{1}_{Gs \ i})(i := \mathbf{1}_{Gs \ i})$ using *i* by force thus $g \ x \ i \ \mathbf{1}_{Gs \ i} \neq 0$ using *i* $g \ DirProds-one''[of \ Gs \ I] \ xc.char-one-nz$ by auto **show** g x i a = 0 if $a: a \notin carrier (Gs i)$ for a proof from a *i* have $((\lambda i \in I. \mathbf{1}_{Gs i})(i := a)) \notin carrier (DirProds Gs I)$ unfolding DirProds-def by force from xc.char-eq-0[OF this] show ?thesis using i g by auto qed show $g x i (a \otimes_{Gs i} b) = g x i a * g x i b$ if ab: $a \in carrier (Gs i) b \in carrier (Gs i)$ for a b proof have $g x i (a \otimes_{Gs i} b)$ $= x \ ((\lambda i \in I. \ \mathbf{1}_{Gs \ i})(i := a) \otimes_{DirProds \ Gs \ I} (\lambda i \in I. \ \mathbf{1}_{Gs \ i})(i := b))$ proof have $((\lambda i \in I. \mathbf{1}_{Gs \ i})(i := a) \otimes_{DirProds \ Gs \ I} (\lambda i \in I. \mathbf{1}_{Gs \ i})(i := b))$ $= ((\lambda i \in I. \mathbf{1}_{Gs \ i})(i := (a \otimes_{Gs \ i} b)))$ proof have $((\lambda i \in I. \mathbf{1}_{Gs \ i})(i := a) \otimes_{DirProds \ Gs \ I} (\lambda i \in I. \mathbf{1}_{Gs \ i})(i := b)) j$ $= ((\lambda i \in I. \mathbf{1}_{Gs i})(i := (a \otimes_{Gs i} b))) j$ for j**proof** (cases $j \in I$) case True from allg[OF True] interpret Gj: group Gs j. show ?thesis using ab True i unfolding DirProds-mult by simp \mathbf{next} case False then show ?thesis unfolding DirProds-mult using i by fastforce qed thus ?thesis by fast qed thus ?thesis using i q by auto \mathbf{qed} also have $\ldots = x ((\lambda i \in I. \mathbf{1}_{Gs \ i})(i := a)) * x ((\lambda i \in I. \mathbf{1}_{Gs \ i})(i := b))$ proof have ac: $((\lambda i \in I. \mathbf{1}_{Gs})(i := a)) \in carrier (DirProds Gs I)$ unfolding DirProds-def using ab i monoid.one-closed[OF group.is-monoid[OF allg]] by force have bc: $((\lambda i \in I. \mathbf{1}_{Gs})(i := b)) \in carrier (DirProds Gs I)$ unfolding DirProds-def using ab i monoid.one-closed[OF group.is-monoid[OF allg]] by force from $xc.char-mult[OF \ ac \ bc]$ show ?thesis. ged also have $\ldots = g x i a * g x i b$ using i g by *auto* finally show ?thesis .

```
qed
  qed
qed
lemma Characters-DirProds-single-prod:
  assumes finite-comm-group (DirProds Gs I)
  and x: x \in carrier (Characters (DirProds Gs I))
  and I: finite I
  defines g: g \equiv (\lambda I. (\lambda c. (\lambda i \in I. (\lambda a. c ((\lambda i \in I. \mathbf{1}_{Gs i})(i:=a))))))
  shows (\lambda e. if e \in carrier(DirProds Gs I) then \prod i \in I. (g I x i) (e i) else \theta) = x
(\mathbf{is} ? g x = x)
proof
  show ?g x e = x e for e
  proof (cases e \in carrier (DirProds Gs I))
   case True
   show ?thesis using I x assms(1) True unfolding q
   proof(induction I arbitrary: x e rule: finite-induct)
     case empty
     interpret DP: finite-comm-group DirProds Gs {} by fact
      from DirProds-empty[of Gs] have order (DirProds Gs {}) = 1 unfolding
order-def by simp
     with DP.characters-in-order-1 [OF this] empty(1) show ?case
        using DirProds-empty[of Gs] unfolding Characters-def principal-char-def
by auto
   \mathbf{next}
     case j: (insert j I)
     interpret DP: finite-comm-group DirProds Gs (insert j I) by fact
     interpret DP2: finite-comm-group DirProds Gs I
     proof -
     from DirProds-finite-comm-group-iff [of insert j I Gs] DP.finite-comm-group-axioms
j
       have (\forall i \in (insert \ j \ I)). finite-comm-group (Gs i)) by blast
     with DirProds-finite-comm-group-iff [OF j(1), of Gs] show finite-comm-group
(DirProds \ Gs \ I)
         by blast
     qed
     interpret xc: character DirProds Gs (insert j I) x
       using j(4) unfolding Characters-def characters-def by simp
     have all g: \bigwedge i. i \in (insert \ j \ I) \Longrightarrow group \ (Gs \ i)
       using DirProds-group-imp-groups[OF DP.is-group].
     have e1c: e(j := \mathbf{1}_{G_s \ j}) \in carrier \ (DirProds \ Gs \ (insert \ j \ I))
       using j(6) monoid.one-closed[OF group.is-monoid[OF allg[of j]]]
       unfolding DirProds-def PiE-def Pi-def by simp
     have e2c: (\lambda i \in (insert \ j \ I), \mathbf{1}_{Gs \ i})(j := e \ j) \in carrier \ (DirProds \ Gs \ (insert \ j
I))
       unfolding DirProds-def PiE-def Pi-def
         using monoid.one-closed[OF group.is-monoid[OF allg]] comp-in-carr[OF
j(6)] by auto
     have e = e(j := \mathbf{1}_{Gs j}) \otimes_{DirProds Gs (insert j I)} (\lambda i \in (insert j I). \mathbf{1}_{Gs i})(j :=
```

17

e jproof have e k $= (e(j:= \mathbf{1}_{Gs \ j}) \otimes_{DirProds \ Gs \ (insert \ j \ I)} (\lambda i \in (insert \ j \ I). \ \mathbf{1}_{Gs \ i})(j:= e$ j)) kfor k**proof**(cases $k \in (insert \ j \ I)$) case k: True from allg[OF k] interpret Gk: group Gs k. **from** allg[of j] **interpret** Gj: group Gs j **by** simp from k show ?thesis unfolding comp-mult[OF k] using comp-in-carr[OF j(6) k] by auto next case False then show ?thesis using j(6) unfolding DirProds-def by auto qed thus ?thesis by blast qed hence $x e = x (e(j := \mathbf{1}_{G_s j})) * x ((\lambda i \in (insert j I), \mathbf{1}_{G_s i})(j := e j))$ using xc.char-mult[OF e1c e2c] by argo also have ... = $(\prod i \in I. g (insert j I) x i (e i)) * g (insert j I) x j (e j)$ proof have $x \ (e(j:=\mathbf{1}_{Gs\ j})) = (\prod i \in I. \ g \ (insert\ j\ I) \ x\ i \ (e\ i))$ proof have eu: $e(j:=undefined) \in carrier (DirProds Gs I)$ using j(2, 6)unfolding DirProds-def PiE-def Pi-def extensional-def by fastforce let $?x = \lambda p$. if $p \in carrier(DirProds Gs I)$ then $x (p(j := \mathbf{1}_{Gs j}))$ else 0 have cx2: character (DirProds Gs I) ?x proof show $?x \mathbf{1}_{DirProds Gs I} \neq 0$ proof have $\mathbf{1}_{DirProds \ Gs \ I}(j := \mathbf{1}_{Gs \ j}) = \mathbf{1}_{DirProds \ Gs \ (insert \ j \ I)}$ unfolding DirProds-one" by force thus ?thesis by simp qed show $?x \ a = 0$ if $a: a \notin carrier$ (DirProds Gs I) for a using a by argo show $?x (a \otimes_{DirProds} G_{s,I} b) = ?x a * ?x b$ if ab: $a \in carrier$ (DirProds Gs I) $b \in carrier$ (DirProds Gs I) for a b proof have ac: $a(j := \mathbf{1}_{Gs \ j}) \in carrier \ (DirProds \ Gs \ (insert \ j \ I))$ **using** *ab monoid.one-closed*[*OF group.is-monoid*[*OF allg*[*of j*]]] unfolding DirProds-def PiE-def Pi-def by simp have bc: $b(j := \mathbf{1}_{Gs \ j}) \in carrier \ (DirProds \ Gs \ (insert \ j \ I))$ using ab monoid.one-closed[OF group.is-monoid[OF allg[of j]]] unfolding DirProds-def PiE-def Pi-def by simp have m: $((a \otimes_{DirProds \ Gs \ I} b)(j := \mathbf{1}_{Gs \ j}))$ $= (a(j := \mathbf{1}_{Gs \ j}) \otimes_{DirProds \ Gs \ (insert \ j \ I)} b(j := \mathbf{1}_{Gs \ j}))$ proof –

have $((a \otimes_{DirProds Gs I} b)(j := \mathbf{1}_{Gs j})) h$ $= (a(j := \mathbf{1}_{Gs \ j}) \otimes_{DirProds \ Gs \ (insert \ j \ I)} b(j := \mathbf{1}_{Gs \ j})) \ h$ if $h: h \in (insert \ j \ I)$ for h**proof**(cases h=j) case True **interpret** G_j : group $G_s j$ using allg[of j] by blast **from** True comp-mult[OF h, of Gs $a(j := \mathbf{1}_{Gs \ j}) \ b(j := \mathbf{1}_{Gs \ j})$] show ?thesis by auto \mathbf{next} case False **interpret** G_j : group $G_s h$ using allg[OF h]. **from** False h comp-mult[OF h, of Gs $a(j := \mathbf{1}_{Gs \ j}) \ b(j := \mathbf{1}_{Gs \ j})$] comp-mult[of h I Gs a b] show ?thesis by auto \mathbf{qed} moreover have $((a \otimes_{DirProds Gs I} b)(j := \mathbf{1}_{Gs j})) h$ $= (a(j := \mathbf{1}_{Gs \ j}) \otimes_{DirProds \ Gs \ (insert \ j \ I)} b(j := \mathbf{1}_{Gs \ j})) \ h$ if h: $h \notin (insert \ j \ I)$ for h using h unfolding DirProds-def PiE-def by simp ultimately show ?thesis by blast qed have $x ((a \otimes_{DirProds Gs I} b)(j := \mathbf{1}_{Gs j}))$ $= x \ (a(j := \mathbf{1}_{Gs \ j})) * x \ (b(j := \mathbf{1}_{Gs \ j}))$ **by** (unfold m, intro xc.char-mult[OF ac bc]) thus ?thesis using ab by auto qed qed then interpret cx2: character DirProds Gs I ?x. from cx2 have cx3:? $x \in carrier$ (Characters (DirProds Gs I)) unfolding Characters-def characters-def by simp from $j(3)[OF \ cx3 \ DP2.finite-comm-group-axioms \ eu]$ have $(if \ e(j:=undefined) \in carrier \ (DirProds \ Gs \ I)$ then $\prod i \in I$. g I ?x i ((e(j:=undefined)) i) else 0) = ?x (e(j:=undefined))using eu j(2) unfolding g by fast with *eu* have $(\prod i \in I. g \ I \ (\lambda p. if p \in carrier \ (DirProds \ Gs \ I))$ then $x (p(j := \mathbf{1}_{Gs j}))$ else 0) i ((e(j := undefined)))) = x (e(j := $\mathbf{1}_{Gs j}))$ by simp **moreover have** $g I (\lambda a. if a \in carrier (DirProds Gs I))$ then $x (a(j := \mathbf{1}_{Gs j}))$ else 0) i ((e(j := undefined)) i) = g (insert j I) x i (e i) if $i: i \in I$ for iproof have $(\lambda i \in I. \mathbf{1}_{Gs i})(i := e i) \in carrier (DirProds Gs I)$

unfolding DirProds-def PiE-def Pi-def extensional-def

```
using monoid.one-closed[OF group.is-monoid[OF allg]] comp-in-carr[OF
j(6)] i by simp
           moreover have ((\lambda i \in I. \mathbf{1}_{Gs \ i})(i := e \ i, j := \mathbf{1}_{Gs \ j}))
                        = ((\lambda i \in insert \ j \ I. \mathbf{1}_{Gs} \ i)(i := e \ i)) using i \ j(2) by auto
           ultimately show ?thesis using i j(2, 4, 6) unfolding g by auto
         qed
         ultimately show ?thesis by simp
       qed
       moreover have x ((\lambda i \in (insert \ j \ I), \mathbf{1}_{Gs \ i})(j := e \ j)) = g (insert \ j \ I) \ x \ j \ (e \ insert \ j \ I) 
j)
         unfolding g by simp
       ultimately show ?thesis by argo
     qed
     finally show ?case using j unfolding g by auto
   ged
  next
   case False
   interpret xc: character DirProds Gs I x
     using x unfolding Characters-def characters-def by simp
   from xc.char-eq-0[OF False] False show ?thesis by argo
  qed
\mathbf{qed}
```

This allows for the following: the character group of a direct product is isomorphic to the direct product of the character groups of the factors.

lemma (in *finite-comm-group*) Characters-DirProds-iso: assumes DirProds Gs $I \cong G$ group (DirProds Gs I) finite I **shows** DirProds (Characters \circ Gs) $I \cong$ Characters G proof interpret DP: group DirProds Gs I by fact interpret DP: finite-comm-group DirProds Gs I by (intro iso-imp-finite-comm[OF DP.iso-sym[OF assms(1)]], unfold-locales) interpret DPC: finite-comm-group DirProds (Characters \circ Gs) I using DirProds-finite-comm-group-iff[OF assms(3), of Characters \circ Gs] DirProds-finite-comm-group-iff [OF assms(3), of Gs] DP.finite-comm-group-axioms finite-comm-group.finite-comm-group-Characters by auto interpret CDP: finite-comm-group Characters (DirProds Gs I) using DP.finite-comm-group-Characters. interpret C: finite-comm-group Characters G using finite-comm-group-Characters have allg: $\bigwedge i. i \in I \Longrightarrow group (Gs i)$ using DirProds-group-imp-groups[OF assms(2)] let $?f = (\lambda cp. (\lambda e. (if e \in carrier (DirProds Gs I) then \prod i \in I. cp i (e i) else 0)))$ have f-in: ?f $x \in carrier$ (Characters (DirProds Gs I)) if $x: x \in carrier$ (DirProds (Characters \circ Gs) I) for x **proof**(unfold carrier-Characters characters-def, safe, unfold-locales) show $?f x \mathbf{1}_{DirProds Gs I} \neq 0$ proof -

20

have $x \ i \ (\mathbf{1}_{DirProds \ Gs \ I} \ i) \neq 0$ if $i: i \in I$ for iproof interpret Gi: finite-comm-group Gs i using DirProds-finite-comm-group-iff[OF assms(3)] DP.finite-comm-group-axioms i **by** blast interpret xi: character Gs i x iusing *i x* unfolding *DirProds-def Characters-def characters-def* by *auto* show ?thesis using DirProds-one'[OF i, of Gs] by simp qed thus ?thesis by $(simp \ add: assms(3))$ qed show ?f x = 0 if $a \notin carrier$ (DirProds Gs I) for a using that by simp **show** $?f x (a \otimes_{DirProds Gs I} b) = ?f x a * ?f x b$ if ab: $a \in carrier$ (DirProds Gs I) $b \in carrier$ (DirProds Gs I) for a b proof – have $a \otimes_{DirProds Gs I} b \in carrier (DirProds Gs I)$ using that by blast moreover have $(\prod i \in I. x i ((a \otimes_{DirProds Gs I} b) i))$ $= (\prod i \in I. \ x \ i \ (a \ i)) * (\prod i \in I. \ x \ i \ (b \ i))$ proof have $x \ i \ ((a \otimes_{DirProds} G_{S \ I} \ b) \ i) = x \ i \ (a \ i) \ * x \ i \ (b \ i)$ if $i: \ i \in I$ for iproof – interpret xi: character $Gs \ i \ x \ i$ using *i x* unfolding *DirProds-def Characters-def characters-def* by *auto* **show** ?thesis using ab comp-mult[OF i, of Gs a b] by(auto simp: comp-in-carr[OF - i])qed thus ?thesis using prod.distrib by force ged ultimately show ?thesis using that by auto qed qed have $?f \in iso (DirProds (Characters \circ Gs) I) (Characters (DirProds Gs I))$ **proof** (*intro isoI*) **show** $?f \in hom (DirProds (Characters \circ Gs) I) (Characters (DirProds Gs I))$ **proof** (*intro homI*) **show** ?f $x \in carrier$ (Characters (DirProds Gs I)) if $x: x \in carrier$ (DirProds (Characters \circ Gs) I) for x using f-in[OF that]. **show** $?f(x \otimes_{DirProds} (Characters \circ Gs) | y) = ?fx \otimes_{Characters} (DirProds Gs I)$?f yif $x \in carrier$ (DirProds (Characters \circ Gs) I) $y \in carrier$ (DirProds $(Characters \circ Gs) I)$ for x yproof have $?f x \otimes_{Characters (DirProds Gs I)} ?f y$ = $(\lambda e. if e \in carrier (DirProds Gs I) then (\prod i \in I. x i (e i)) * (\prod i \in I. y i)$ (e i) else 0unfolding Characters-def by auto also have $\ldots = ?f (x \otimes_{DirProds} (Characters \circ Gs) I y)$ proof -

have $(\prod i \in I. x i (e i)) * (\prod i \in I. y i (e i))$ $= (\prod i \in I. (x \otimes_{DirProds} (Characters \circ Gs) I y) i (e i))$ for e unfolding DirProds-def Characters-def by (auto simp: prod.distrib) thus ?thesis by presburger qed finally show ?thesis by argo qed qed then interpret fgh: group-hom DirProds (Characters \circ Gs) I Characters (DirProds Gs I) ?f **by** (*unfold-locales*, *simp*) **show** bij-betw ?f (carrier (DirProds (Characters \circ Gs) I)) (carrier (Characters (DirProds Gs I))) **proof** (*intro bij-betwI*) $\textbf{let } ?g = (\lambda c. \ (\lambda i \in I. \ (\lambda a. \ c \ ((\lambda i \in I. \ \textbf{1}_{Gs \ i})(i{:=}a)))))$ have allc: character (Gs i) (?g x i) if $x: x \in carrier$ (Characters (DirProds Gs I)) and $i: i \in I$ for x iusing DirProds-subchar[OF DP.finite-comm-group-axioms x i assms(3)]. have g-in: $?g x \in carrier (DirProds (Characters \circ Gs) I)$ if $x: x \in carrier$ (Characters (DirProds Gs I)) for x using *allc*[OF x] unfolding DirProds-def Characters-def characters-def by simp **show** fi: $?f \in carrier$ (DirProds (Characters \circ Gs) I) \rightarrow carrier (Characters $(DirProds \ Gs \ I))$ using *f*-in by *fast* show gi: $?g \in carrier$ (Characters (DirProds Gs I)) \rightarrow carrier (DirProds $(Characters \circ Gs) I)$ using *g*-in by fast show ?f (?g x) = x if x: $x \in carrier$ (Characters (DirProds Gs I)) for x proof – from x interpret x: character DirProds Gs I x unfolding Characters-def characters-def by *auto* **from** f-in[OF g-in[OF x]] **interpret** character DirProds Gs I ?f (?g x) unfolding Characters-def characters-def by simp have $(\prod i \in I. (\lambda i \in I. \lambda a. x ((\lambda i \in I. \mathbf{1}_{Gs i})(i := a))) i (e i)) = x e$ if $e: e \in carrier$ (DirProds Gs I) for eproof define y where y: $y = (\lambda e. if e \in carrier (DirProds Gs I))$ then $\prod i \in I$. $(\lambda i \in I. \lambda a. x ((\lambda i \in I. \mathbf{1}_{G_s} i)(i := a))) i$ (e i)else 0) **from** Characters-DirProds-single-prod[OF DP.finite-comm-group-axioms x assms(3)] have y = x using y by force hence y e = x e by blast thus ?thesis using e unfolding y by argo

ged

with x.char-eq-0 show ?thesis by force

\mathbf{qed}

show ?g(?fx) = x if $x: x \in carrier$ (DirProds (Characters \circ Gs) I) for x $proof(intro \ eq-parts-imp-eq[OF \ g-in[OF \ f-in[OF \ x]] \ x])$ show ?g(?fx) i = x i if $i: i \in I$ for iproof interpret xi: character $Gs \ i \ x \ i$ using x i unfolding DirProds-def Characters-def characters-def by auto have ?g(?fx) i a = x i a if $a: a \notin carrier(Gs i)$ for a proof have $(\lambda i \in I. \mathbf{1}_{Gs})(i := a) \notin carrier (DirProds Gs I)$ using a *i* unfolding *DirProds-def PiE-def Pi-def* by *auto* with xi.char-eq-0[OF a] a i show ?thesis by auto qed **moreover have** ?g (?f x) i = x i a if $a: a \in carrier$ (Gs i) for a proof have $(\lambda i \in I. \mathbf{1}_{Gs \ i})(i := a) \in carrier \ (DirProds \ Gs \ I)$ using a *i* monoid.one-closed[OF group.is-monoid[OF allg]] unfolding DirProds-def by force moreover have $(\prod j \in I. x j (((\lambda i \in I. \mathbf{1}_{Gs i})(i := a)) j)) = x i a$ proof – $\begin{array}{l} \mathbf{have} \; (\prod j \in I. \; x \; j \; (((\lambda i \in I. \; \mathbf{1}_{Gs \; i})(i := a)) \; j)) \\ = \; x \; i \; (((\lambda i \in I. \; \mathbf{1}_{Gs \; i})(i := a)) \; i) \; \ast \; (\prod j \in I - \{i\}. \; x \; j \; (((\lambda i \in I. \; \mathbf{1}_{Gs \; i})(i = a)) \; i)) \end{array}$:= a)) j))**by** (meson assms(3) i prod.remove) moreover have x j ((($\lambda i \in I$. $\mathbf{1}_{G_s} i$)(i := a)) j) = 1 if $j: j \in I j \neq i$ for jproof – **interpret** xj: character Gs j x jusing j(1) x unfolding DirProds-def Characters-def characters-def by auto show ?thesis using j by auto qed moreover have $x i (((\lambda i \in I. \mathbf{1}_{Gs} i)(i := a)) i) = x i a$ by simp ultimately show ?thesis by auto qed ultimately show ?thesis using a i by simp qed ultimately show ?thesis by blast qed qed qed qed hence DirProds (Characters \circ Gs) $I \cong$ Characters (DirProds Gs I) unfolding is-iso-def by blast **moreover have** Characters (DirProds Gs I) \cong Characters G using DP.iso-imp-iso-chars[OF assms(1) is-group]. ultimately show ?thesis using iso-trans by blast qed

As thus both the group and its character group can be decomposed into the

same cyclic factors, the isomorphism follows for any finite abelian group.

theorem (in *finite-comm-group*) Characters-iso: **shows** $G \cong$ Characters G proof – from cyclic-product obtain ns where ns: DirProds $(\lambda n. Z (ns ! n)) \{... < length ns\} \cong G \forall n \in set ns. n \neq 0$. **interpret** DP: group DirProds $(\lambda n. Z (ns ! n)) \{..< length ns\}$ **by** (*intro* DirProds-is-group, auto) have $G \cong DirProds$ ($\lambda n. Z$ (ns ! n)) {...< length ns} using DP.iso-sym[OF ns(1)] **moreover have** DirProds (Characters \circ (λn . Z (ns ! n))) {...<length ns} \cong Characters G by (intro Characters-DirProds-iso[OF ns(1) DirProds-is-group], auto) moreover have DirProds ($\lambda n. Z$ (ns ! n)) {...<length ns} \cong DirProds (Characters \circ ($\lambda n. Z$ (ns ! n))) {...<length ns} proof (intro DirProds-iso1) fix *i* assume *i*: $i \in \{..< length ns\}$ obtain a where cyclic-group (Z(ns!i)) a using Zn-cyclic-group. then interpret Zi: cyclic-group Z(ns!i) a. interpret Zi: finite-cyclic-group Z (ns!i) a proof have order $(Z (ns ! i)) \neq 0$ using ns(2) i Zn-order by simp thus finite (carrier (Z(ns!i))) unfolding order-def by (simp add: card-eq-0-iff) ged **show** Group.group ((Characters \circ ($\lambda n. Z$ (ns ! n))) i) Group.group $(Z (ns ! i)) Z (ns ! i) \cong (Characters \circ (\lambda n. Z (ns ! n))) i$ using Zi. Characters-iso Zi. finite-comm-group-Characters comm-group-def finite-comm-group-def by auto qed ultimately show ?thesis by (auto elim: iso-trans) qed

Hence, the orders are also equal.

corollary (in finite-comm-group) order-Characters: order (Characters G) = order Gusing iso-same-card[OF Characters-iso] unfolding order-def by argo

corollary (in *finite-comm-group*) card-characters: card (characters G) = order Gusing order-Characters unfolding order-def Characters-def by simp

1.5 Non-trivial facts about characters

We characterize the character group of a quotient group as the group of characters that map all elements of the subgroup onto 1.

lemma (in finite-comm-group) iso-Characters-FactGroup: assumes H: subgroup H Gshows ($\lambda \chi x$. if $x \in carrier G$ then $\chi (H \#> x)$ else θ) \in

```
iso (Characters (G Mod H)) ((Characters G)(carrier := \{\chi \in characters
G. \forall x \in H. \chi x = 1 \}
proof -
 interpret H: normal H G using subgroup-imp-normal[OF H].
 interpret Chars: finite-comm-group Characters G
   by (rule finite-comm-group-Characters)
  interpret Fact: comm-group G Mod H
  by (simp add: H.subgroup-axioms comm-group.abelian-FactGroup comm-group-axioms)
  interpret Fact: finite-comm-group G Mod H
   by unfold-locales (auto simp: carrier-FactGroup)
 define C :: ('a \Rightarrow complex) set where C = \{\chi \in characters \ G. \ \forall x \in H. \ \chi \ x = 1\}
 interpret C: subgroup C Characters G
 proof (unfold-locales, goal-cases)
   case 1
   thus ?case
    by (auto simp: C-def one-Characters mult-Characters carrier-Characters char-
acters-def)
 \mathbf{next}
   case 2
   thus ?case
    by (auto simp: C-def one-Characters mult-Characters carrier-Characters char-
acters-def)
  \mathbf{next}
   case 3
   thus ?case
     by (auto simp: C-def one-Characters mult-Characters
                  carrier-Characters characters-def principal-char-def)
  next
   case (4 \chi)
   hence inv_{Characters \ G} \ \chi = inv-character \chi
     by (subst inv-Characters') (auto simp: C-def carrier-Characters)
   moreover have inv-character \chi \in characters G
     using 4 by (auto simp: C-def characters-def)
   moreover have \forall x \in H. inv-character \chi x = 1
     using 4 by (auto simp: C-def inv-character-def)
   ultimately show ?case
     by (auto simp: C-def)
 qed
  define f :: ('a \ set \Rightarrow complex) \Rightarrow ('a \Rightarrow complex)
   where f = (\lambda \chi \ x. \ if \ x \in carrier \ G \ then \ \chi \ (H \ \#>x) \ else \ 0)
 have [intro]: character G (f \chi) if character (G Mod H) \chi for \chi
  proof –
   interpret character G Mod H \chi by fact
   show ?thesis
   proof (unfold-locales, goal-cases)
     case 1
```

```
thus ?case by (auto simp: f-def char-eq-0-iff carrier-FactGroup)
 \mathbf{next}
   case (2 x)
   thus ?case by (auto simp: f-def)
 next
   case (3 x y)
   have \chi (H #> x) * \chi (H #> y) = \chi ((H #> x) \otimes_{G Mod H} (H #> y))
     using 3 by (intro char-mult [symmetric]) (auto simp: carrier-FactGroup)
   also have (H \#> x) \otimes_G Mod H (H \#> y) = H \#> (x \otimes y)
     using 3 by (simp add: H.rcos-sum)
   finally show ?case
    using 3 by (simp add: f-def)
 qed
qed
have [intro]: f \ \chi \in C if character (G Mod H) \chi for \chi
proof -
 interpret \chi: character G Mod H \chi
   by fact
 have character G (f \chi)
   using \chi.character-axioms by auto
 moreover have \chi (H \# > x) = 1 if x \in H for x
   using that H.rcos-const \chi.char-one by force
 ultimately show ?thesis
   by (auto simp: carrier-Characters C-def characters-def f-def)
qed
show f \in iso (Characters (G Mod H)) ((Characters G)(carrier := C))
proof (rule isoI)
 show f \in hom (Characters (G Mod H)) (Characters G(carrier := C))
 proof (rule homI, goal-cases)
   case (1 \chi)
   thus ?case
    by (auto simp: carrier-Characters characters-def)
 qed (auto simp: f-def carrier-Characters fun-eq-iff mult-Characters)
\mathbf{next}
 have bij-betw f (characters (G Mod H)) C
   unfolding bij-betw-def
 proof
   show inj: inj-on f (characters (G \mod H))
   proof (rule inj-onI, goal-cases)
    case (1 \chi 1 \chi 2)
    interpret \chi_1: character G Mod H \chi_1
      using 1 by (auto simp: characters-def)
    interpret \chi 2: character G Mod H \chi 2
      using 1 by (auto simp: characters-def)
    have \chi 1 H' = \chi 2 H' for H'
    proof (cases H' \in carrier (G Mod H))
```

```
26
```

```
case False
        thus ?thesis by (simp add: \chi 1.char-eq-0 \ \chi 2.char-eq-0)
       \mathbf{next}
        case True
        then obtain x where x: x \in carrier \ G \ H' = H \ \# > x
          by (auto simp: carrier-FactGroup)
        from 1 have f \chi 1 x = f \chi 2 x
          by simp
        with x show ?thesis
          by (auto simp: f-def)
       \mathbf{qed}
       thus \chi 1 = \chi 2 by force
     qed
     have f ' characters (G Mod H) \subseteq C
       by (auto simp: characters-def)
     moreover have C \subseteq f ' characters (G Mod H)
     proof safe
       fix \chi assume \chi: \chi \in C
       from \chi interpret character G \chi
        by (auto simp: C-def characters-def)
      have [simp]: \chi x = 1 if x \in H for x
        using \chi that by (auto simp: C-def)
      have \forall H' \in carrier (G Mod H). \exists x \in carrier G. H' = H \# > x
        by (auto simp: carrier-FactGroup)
      then obtain h where h: h H' \in carrier \ G H' = H \ \# > h H'  if H' \in carrier
(G Mod H) for H'
        by metis
      define \chi' where \chi' = (\lambda H') if H' \in carrier (G Mod H) then \chi (h H') else
\theta)
       have \chi-cong: \chi x = \chi y if H \# > x = H \# > y x \in carrier G y \in carrier
G for x y
       proof -
        have x \in H \# > x
          by (simp add: H.subgroup-axioms rcos-self that (2))
        also have \ldots = H \# > y
          by fact
        finally obtain z where z: z \in H x = z \otimes y
          unfolding r-coset-def by auto
        thus ?thesis
          using z H.subset that by simp
       qed
       have character (G Mod H) \chi'
       proof (unfold-locales, goal-cases)
        case 1
        have H: H \in carrier (G Mod H)
```

```
using Fact.one-closed unfolding one-FactGroup.
        with h[of H] have h H \in carrier G
         by blast
        thus ?case using H
         by (auto simp: char-eq-0-iff \chi'-def)
      \mathbf{next}
        case (2 H')
        thus ?case by (auto simp: \chi'-def)
      \mathbf{next}
        case (3 H1 H2)
        from 3 have H12: H1 \langle \# \rangle H2 \in carrier (G Mod H)
          using Fact.m-closed by force
        have \chi (h (H1 <#> H2)) = \chi (h H1 \otimes h H2)
        proof (rule \chi-cong)
         show H \# > h (H1 < \# > H2) = H \# > (h H1 \otimes h H2)
           by (metis 3 H.rcos-sum H12 h)
        qed (use 3 h[of H1] h[of H2] h[OF H12] in auto)
        thus ?case
          using 3 H12 h[of H1] h[of H2] by (auto simp: \chi'-def)
      qed
      moreover have f \chi' x = \chi x for x
      proof (cases x \in carrier G)
        case False
        thus ?thesis
          by (auto simp: f-def \chi'-def char-eq-0-iff)
      \mathbf{next}
        case True
        hence *: H \# > x \in carrier (G Mod H)
         by (auto simp: carrier-FactGroup)
        have \chi (h (H \#> x)) = \chi x
          using True *h[of H \# > x] by (intro \chi-cong) auto
        thus ?thesis
          using True * by (auto simp: f-def fun-eq-iff \chi'-def)
      qed
      hence f \chi' = \chi by force
      ultimately show \chi \in f ' characters (G Mod H)
        unfolding characters-def by blast
     qed
     ultimately show f ' characters (G Mod H) = C
      by blast
   qed
  thus bij-betw f (carrier (Characters (G Mod H))) (carrier (Characters G(carrier
:= C[))
     by (simp add: carrier-Characters)
```

qed

qed

lemma (in finite-comm-group) is-iso-Characters-FactGroup: assumes H: subgroup H G

shows Characters (G Mod H) \cong (Characters G)(carrier := { $\chi \in characters G$. $\forall x \in H. \chi x = 1$ })

using iso-Characters-FactGroup[OF assms] unfolding is-iso-def by blast

In order to derive the number of extensions a character on a subgroup has to the entire group, we introduce the group homomorphism *restrict-char* that restricts a character to a given subgroup H.

definition restrict-char::'a set \Rightarrow ('a \Rightarrow complex) \Rightarrow ('a \Rightarrow complex) where restrict-char H $\chi = (\lambda e. if e \in H then \chi e else 0)$

lemma (in *finite-comm-group*) restrict-char-hom:

assumes subgroup H G shows group-hom (Characters G) (Characters (G((carrier := H)))) (restrict-char H) proof – let ?CG = Characters Glet ?H = G(|carrier := H|)let ?CH = Characters ?H**interpret** *H*: subgroup *H G* **by** fact **interpret** *H*: finite-comm-group ?H **by** (simp add: assms subgroup-imp-finite-comm-group) interpret CG: finite-comm-group ?CG using finite-comm-group-Characters. interpret CH: finite-comm-group ?CH using H.finite-comm-group-Characters. show ?thesis proof(unfold-locales, intro homI) show restrict-char $H x \in carrier ?CH$ if $x: x \in carrier ?CG$ for xproof – interpret xc: character G x using x unfolding Characters-def characters-def by simp have character ?H (restrict-char H x) **by** (unfold restrict-char-def, unfold-locales, auto) thus ?thesis unfolding Characters-def characters-def by simp qed show restrict-char $H(x \otimes_{?CG} y) = restrict-char H x \otimes_{?CH} restrict-char H y$ if $x: x \in carrier ?CG$ and $y: y \in carrier ?CG$ for x yproof interpret xc: character G x using x unfolding Characters-def characters-def by simp interpret yc: character G y using y unfolding Characters-def characters-def by simp show ?thesis unfolding Characters-def restrict-char-def by auto qed \mathbf{qed} qed

The kernel is just the set of the characters that are 1 on all of H.

lemma (in finite-comm-group) restrict-char-kernel: **assumes** subgroup H G **shows** kernel (Characters G) (Characters (G(arrier := H))) (restrict-char H) $= \{\chi \in characters \ G. \ \forall x \in H. \ \chi \ x = 1\}$ **by** (unfold restrict-char-def kernel-def one-Characters carrier-Characters principal-char-def characters-def, simp, metis)

Also, all of the characters on the subgroup are the image of some character on the whole group.

lemma (in *finite-comm-group*) restrict-char-image: assumes subgroup H Gshows restrict-char H (carrier (Characters G)) = carrier (Characters (G) carrier := H)) proof **interpret** H: subgroup H G by fact **interpret** H: finite-comm-group G(carrier := H) using subgroup-imp-finite-comm-group [OF]assms]. **interpret** r: group-hom Characters G Characters (G([carrier := H])) restrict-char Η using restrict-char-hom[OF assms]. interpret Mod: finite-comm-group G Mod H using finite-comm-FactGroup[OF] assms]. interpret CG: finite-comm-group Characters G using finite-comm-group-Characters have c1: order (Characters (G(carrier := H))) = card H using H.order-Characters unfolding order-def by simp have card H * card (kernel (Characters G) (Characters (G(carrier := H))) (restrict-char H)) = order Gusing restrict-char-kernel[OF assms] iso-same-card[OF is-iso-Characters-FactGroup[OF assms]] Mod.order-Characters lagrange[OF assms] unfolding order-def Fact-Group-def **by** (force simp: algebra-simps) **moreover have** card (kernel (Characters G) (Characters (G(carrier := H))) $(restrict-char H)) \neq 0$ using r.one-in-kernel unfolding kernel-def CG.fin by auto ultimately have c2: card H = card (restrict-char H ' carrier (Characters G)) using r.image-kernel-product[unfolded order-Characters] by (metis mult-right-cancel) have restrict-char H (carrier (Characters G)) \subseteq carrier (Characters (G)(carrier := H())by auto with c2 H.fin show ?thesis by (auto, metis H.finite-imp-card-positive c1 card-subset-eq fin-gen order-def r.H.order-gt-0-iff-finite) qed

It follows that any character on H can be extended to a character on G.

lemma (in finite-comm-group) character-extension-exists: assumes subgroup H G character (G(carrier := H)) χ obtains χ' where character $G \chi'$ and $\bigwedge x. x \in H \Longrightarrow \chi' x = \chi x$ proof – from restrict-char-image[OF assms(1)] assms(2) obtain χ' where chi': restrict-char $H \chi' = \chi$ character $G \chi'$ **by** (force simp: carrier-Characters characters-def) thus ?thesis using that restrict-char-def by metis qed For two characters on a group G the number of characters on subgroup Hthat share the values with them is the same for both. **lemma** (in finite-comm-group) character-restrict-card: assumes subgroup H G character G a character G bshows card { $\chi' \in characters G. \forall x \in H. \chi' x = a x$ } = card { $\chi' \in characters G.$ $\forall x \in H. \ \chi' x = b x \}$ proof **interpret** H: subgroup H G by fact **interpret** *H*: finite-comm-group G((carrier := H)) using assms(1)**by** (*simp add: subgroup-imp-finite-comm-group*) interpret CG: finite-comm-group Characters G using finite-comm-group-Characters **interpret** a: character G a by fact **interpret** b: character G b by fact have ac: $a \in carrier$ (Characters G) unfolding Characters-def characters-def using assms by simp have bc: $b \in carrier$ (Characters G) unfolding Characters-def characters-def using assms by simp define f where f: $f = (\lambda c. \ b \otimes_{Characters} G \ inv_{Characters} \ G \ a \otimes_{Characters} \ G \ c)$ **define** g where g: $g = (\lambda c. \ a \otimes_{Characters} G \ inv_{Characters} \ G \ b \otimes_{Characters} G \ c)$ let $?A = \{\chi' \in characters G. \forall x \in H. \chi' x = a x\}$ let $?B = \{\chi' \in characters G. \forall x \in H. \chi' x = b x\}$ have bij-betw f ?A ?Bproof(intro bij-betwI[of - - - q])show $f \in ?A \rightarrow ?B$ proof show $f x \in ?B$ if $x: x \in ?A$ for xproof – interpret xc: character G x using x unfolding characters-def by blast have $xc: x \in carrier$ (Characters G) using x unfolding Characters-def by simp have f x y = b y if $y: y \in H$ for yproof have $(inv_{Characters G} a) y * a y = 1$ by (simp add: a.inv-Characters a.mult-inv-character mult.commute principal-char-def ythus ?thesis unfolding f mult-Characters using x y by fastforce qed

thus $f x \in PB$ unfolding f carrier-Characters[symmetric] using ac bc xc by blast qed qed show $q \in ?B \rightarrow ?A$ proof show $g x \in ?A$ if $x: x \in ?B$ for xproof – interpret xc: character G x using x unfolding characters-def by blast have $xc: x \in carrier$ (Characters G) using x unfolding Characters-def by simp have g x y = a y if $y: y \in H$ for yproof have $(inv_{Characters} G b) y * x y = 1$ using x yby (simp add: b.inv-Characters b.mult-inv-character mult.commute *principal-char-def*) thus ?thesis unfolding q mult-Characters by simp qed thus $g \ x \in ?A$ unfolding g carrier-Characters[symmetric] using ac bc xc by blast qed qed show g(f x) = x if $x: x \in ?A$ for xproof have $xc: x \in carrier$ (Characters G) using x unfolding Characters-def by force with ac bc show ?thesis unfolding f g**by** (*auto simp: CG.m-assoc[symmetric*], metis CG.inv-closed CG.inv-comm CG.l-inv CG.m-assoc CG.r-one) qed show f(g x) = x if $x: x \in ?B$ for xproof have $xc: x \in carrier$ (Characters G) using x unfolding Characters-def by force with ac bc show ?thesis unfolding f g**by** (*auto simp: CG.m-assoc*[*symmetric*], metis CG.inv-closed CG.inv-comm CG.l-inv CG.m-assoc CG.r-one) qed qed thus ?thesis using bij-betw-same-card by blast qed These lemmas allow to show that the number of extensions of a character on H to a character on G is just |G|/|H|.

theorem (in finite-comm-group) card-character-extensions: assumes subgroup H G character (G([carrier := H])) χ shows card { $\chi' \in$ characters G. $\forall x \in H. \chi' x = \chi x$ } * card H = order G proof – interpret H: subgroup H G by fact

32

interpret *H*: finite-comm-group G([carrier := H])using subgroup-imp-finite-comm-group $[OF \ assms(1)]$. **interpret** chi: character $G((carrier := H)) \chi$ by fact interpret C: finite-comm-group Characters G using finite-comm-group-Characters interpret Mod: finite-comm-group G Mod H using finite-comm-FactGroup[OF] assms(1)]. **obtain** a where $a: a \in carrier$ (Characters G) restrict-char $H = \chi$ proof – have $\exists a \in carrier$ (Characters G). restrict-char $H = \chi$ using restrict-char-image[OF assms(1)] assms(2)unfolding carrier-Characters characters-def image-def by force thus ?thesis using that by blast qed show ?thesis proof have p: { $\chi \in characters \ G. \ \forall x \in H. \ \chi \ x = 1$ } = { $\chi \in characters \ G. \ \forall x \in H. \ \chi \ x$ = principal-char G xunfolding principal-char-def by force have ac: $\{\chi' \in characters \ G. \ \forall x \in H. \ \chi' \ x = \chi \ x\} = \{\chi' \in characters \ G. \ \forall x \in H.$ $\chi' x = a x \}$ using a(2) unfolding restrict-char-def by force have card { $\chi \in characters G. \forall x \in H. \chi x = 1$ } = card { $\chi' \in characters G. \forall x \in H.$ $\chi' x = \chi x \}$ by (unfold ac p; intro character-restrict-card[OF assms(1)], use a [unfolded Characters-def characters-def] in auto) **moreover have** card { $\chi \in characters G. \forall x \in H. \chi x = 1$ } = card (carrier (G) Mod H) using *iso-same-card*[OF *is-iso-Characters-FactGroup*[OF *assms*(1)]] Mod.order-Characters[unfolded order-def] by force **moreover have** card (carrier (G Mod H)) * card H = order Gusing lagrange[OF assms(1)] unfolding FactGroup-def by simp ultimately show ?thesis by argo qed qed

Lastly, we can also show that for each $x \in H$ of order n > 1 and each *n*-th root of unity *z*, there exists a character χ on *G* such that $\chi(x) = z$.

lemma (in group) powi-get-exp-self: fixes z::complex assumes $z \cap n = 1 \ x \in carrier \ G \ ord \ x = n \ n > 1$ shows z powi get-exp x x = z proof from assms have ngt0: n > 0 by simp from powi-mod[OF assms(1) ngt0, of get-exp x x] get-exp-self[OF assms(2), unfolded assms(3)] have z powi get-exp x x = z powi (1 mod int n) by argo also have ... = z using assms(4) by simp finally show ?thesis .

\mathbf{qed}

corollary (in finite-comm-group) character-with-value-exists: assumes $x \in carrier \ G$ and $x \neq 1$ and $z \cap ord \ x = 1$ obtains χ where character $G \chi$ and $\chi x = z$ proof **interpret** H: subgroup generate $G \{x\} G$ using generate-is-subgroup assms(1) by simp **interpret** H: finite-comm-group $G([carrier := generate \ G \ \{x\}))$ using subgroup-imp-finite-comm-group[OF H.subgroup-axioms]. **interpret** H: finite-cyclic-group $G((carrier := generate \ G \ \{x\})) x$ **proof**(*unfold finite-cyclic-group-def, safe*) show finite-group ($G((carrier := generate \ G \ \{x\}))$ by unfold-locales **show** cyclic-group ($G([carrier := generate \ G \ \{x\}])$) x **proof**(*intro* H.cyclic-groupI0) show $x \in carrier$ (G(carrier := generate G {x})) using generate.incl[of x $\{x\} \ G$ by simp show carrier $(G((carrier := generate \ G \ \{x\})) = generate \ (G((carrier := generate \ G \ \{x\}))) = generate \ (G((carrier := generate \ G \ \{x$ generate $G \{x\}$) $\{x\}$ using generate-consistent [OF generate-sincl H.subgroup-axioms] by simp qed qed have ox: H.ord x = ord x using H.gen-closed H.subgroup-axioms subgroup-ord-eq by *auto* have ogt1: ord x > 1 using ord-pos by (metis assms(1, 2) less-one nat-neq-iff ord-eq-1) **from** assms H.unity-root-induce-char[unfolded H.ord-gen-is-group-order[symmetric] ox. OF assms(3)] **obtain** c where c: character (G(carrier := generate G $\{x\}$)) c $c = (\lambda a. if a \in carrier (G(carrier := generate G \{x\}))$ then z powi H.get-exp x a else 0) by blast have cx: c x = z unfolding c(2)using *H.powi-get-exp-self*[OF assms(3) - ox ogt1] generate-sincl[of $\{x\}$] by simp **obtain** f where f: character G f $\bigwedge y$. $y \in (generate G \{x\}) \Longrightarrow f y = c y$ using character-extension-exists [OF H.subgroup-axioms c(1)] by blast **show** ?thesis by (intro that [OF f(1)], use cx f(2) generate-sincl in blast) qed In particular, for any x that is not the identity element, there exists a character χ such that $\chi(x) \neq 1$. **corollary** (in *finite-comm-group*) character-neg-1-exists:

assumes $x \in carrier \ G$ and $x \neq 1$ obtains χ where character $G \chi$ and $\chi x \neq 1$ proof – define z where z = cis (2 * pi / ord x)have z-pow-h: $z \ ord x = 1$ by (auto simp: z-def DeMoivre)

from assms have ord $x \ge 1$ by (intro ord-ge-1) auto **moreover have** ord $x \neq 1$ using pow-ord-eq-1 [of x] assms fin by (intro notI) simp-all ultimately have ord x > 1 by linarith have [simp]: $z \neq 1$ proof assume z = 1have bij-betw (λk . cis (2 * pi * real k / real (ord x))) {..<ord x} {z. z \cap ord x} = 1using $\langle ord \ x > 1 \rangle$ by (intro bij-betw-roots-unity) auto hence inj: inj-on $(\lambda k. \ cis \ (2 * pi * real \ k \ / real \ (ord \ x))) \{..< ord \ x\}$ **by** (*auto simp: bij-betw-def*) have $\theta = (1 :: nat)$ using $\langle z = 1 \rangle$ and $\langle ord x > 1 \rangle$ by (intro inj-onD[OF inj]) (auto simp: z-def) thus False by simp qed obtain χ where character $G \chi$ and $\chi x = z$ using character-with-value-exists [OF assms z-pow-h]. **thus** ?thesis using that [of χ] by simp

\mathbf{qed}

1.6 The first orthogonality relation

The entries of any non-principal character sum to 0.

theorem (in character) sum-character: $(\sum x \in carrier \ G. \ \chi \ x) = (if \ \chi = principal-char \ G \ then \ of-nat \ (order \ G) \ else \ \theta)$ **proof** (cases $\chi = principal-char G$) case True hence $(\sum x \in carrier \ G. \ \chi \ x) = (\sum x \in carrier \ G. \ 1)$ by (intro sum.cong) (auto simp: principal-char-def) also have $\ldots = order \ G \ by \ (simp \ add: order-def)$ finally show ?thesis using True by simp \mathbf{next} case False define S where $S = (\sum x \in carrier \ G, \ \chi \ x)$ from False obtain y where y: $y \in carrier \ G \ \chi \ y \neq 1$ by (auto simp: principal-char-def fun-eq-iff char-eq-0-iff split: if-splits) from y have $S = (\sum x \in carrier \ G. \ \chi \ (y \otimes x))$ unfolding S-def **by** (*intro sum.reindex-bij-betw* [*symmetric*] *bij-betw-mult-left*) also have $\ldots = (\sum x \in carrier \ G. \ \chi \ y * \chi \ x)$ by (intro sum.cong refl char-mult y) also have $\ldots = \chi \ y \ast S$ by (simp add: S-def sum-distrib-left) finally have $(\chi \ y - 1) * S = 0$ by (simp add: algebra-simps) with y have S = 0 by simpwith False show ?thesis by (simp add: S-def) qed

corollary (in *finite-comm-group*) character-orthogonality1: assumes character G χ and character G χ' shows $(\sum x \in carrier \ G. \ \chi \ x \ * \ cnj \ (\chi' \ x)) = (if \ \chi = \chi' \ then \ of -nat \ (order \ G))$ else 0) proof define C where [simp]: C = Characters G interpret C: finite-comm-group C unfolding C-def **by** (*rule finite-comm-group-Characters*) let $?\chi = \lambda x. \ \chi \ x * inv-character \ \chi' \ x$ **interpret** character $G \lambda x. \chi x * inv$ -character $\chi' x$ **by** (*intro character-mult character.inv-character assms*) have $(\sum x \in carrier \ G. \ \chi \ x * cnj \ (\chi' \ x)) = (\sum x \in carrier \ G. \ ?\chi \ x)$ $\mathbf{by}~(intro~sum.cong)~(auto~simp:~inv-character-def)$ also have ... = (if $?\chi$ = principal-char G then of-nat (order G) else 0) by (rule sum-character) also have $?\chi = principal-char \ G \longleftrightarrow \chi \otimes_C inv_C \ \chi' = \mathbf{1}_C$ using assms by (simp add: Characters-simps characters-def) also have $\ldots \longleftrightarrow \chi = \chi'$ proof assume $\chi \otimes_C inv_C \chi' = \mathbf{1}_C$ from C.inv-equality [OF this] and assms show $\chi = \chi'$ **by** (*auto simp: characters-def Characters-simps*) \mathbf{next} assume *: $\chi = \chi'$ from assms show $\chi \otimes_C inv_C \chi' = \mathbf{1}_C$ by (subst *, intro C.r-inv) (auto simp: carrier-Characters characters-def) ged finally show ?thesis . qed

1.7 The isomorphism between a group and its double dual

Lastly, we show that the double dual of a finite abelian group is naturally isomorphic to the original group via the obvious isomorphism $x \mapsto (\chi \mapsto \chi(x))$. It is easy to see that this is a homomorphism and that it is injective. The fact $|\hat{\hat{G}}| = |\hat{G}| = |G|$ then shows that it is also surjective.

context finite-comm-group begin

definition double-dual-iso :: $a \Rightarrow (a \Rightarrow complex) \Rightarrow complex$ where double-dual-iso $x = (\lambda \chi. \text{ if character } G \chi \text{ then } \chi x \text{ else } 0)$

lemma double-dual-iso-apply [simp]: character $G \chi \Longrightarrow$ double-dual-iso $x \chi = \chi x$ by (simp add: double-dual-iso-def)

lemma character-double-dual-iso [intro]: **assumes** $x: x \in carrier \ G$ **shows** character (Characters G) (double-dual-iso x)
proof interpret G': finite-comm-group Characters G
by (rule finite-comm-group-Characters)
show character (Characters G) (double-dual-iso x)
using x by unfold-locales (auto simp: double-dual-iso-def characters-def
characters-def

principal-char-def character.char-eq-0)

qed

lemma double-dual-iso-mult [simp]: assumes $x \in carrier \ G \ y \in carrier \ G$ shows double-dual-iso $(x \otimes y) =$ double-dual-iso $x \otimes_{Characters (Characters G)} double-dual-iso y$ using assms by (auto simp: double-dual-iso-def Characters-def fun-eq-iff charac*ter.char-mult*) **lemma** double-dual-iso-one [simp]: double-dual-iso $\mathbf{1} = principal-char$ (Characters G) by (auto simp: fun-eq-iff double-dual-iso-def principal-char-def carrier-Characters characters-def character.char-one) lemma inj-double-dual-iso: inj-on double-dual-iso (carrier G) proof interpret G': finite-comm-group Characters G **by** (*rule finite-comm-group-Characters*) **interpret** G'': finite-comm-group Characters (Characters G) **by** (rule G'.finite-comm-group-Characters) have hom: double-dual-iso \in hom G (Characters (Characters G)) by (rule homI) (auto simp: carrier-Characters characters-def) have inj-aux: x = 1if $x: x \in carrier \ G \ double-dual-iso \ x = \mathbf{1}_{Characters \ (Characters \ G)}$ for x**proof** (*rule ccontr*) assume $x \neq 1$ **obtain** χ where χ : character $G \chi \chi x \neq 1$ using character-neq-1-exists [OF $x(1) \langle x \neq 1 \rangle$]. **from** x have $\forall \chi$. (if $\chi \in characters G$ then $\chi x else 0$) = (if $\chi \in characters$ G then 1 else 0) $\mathbf{by} \ (auto \ simp: \ double-dual-iso-def \ Characters-def \ fun-eq-iff$ principal-char-def characters-def) hence eq1: $\forall \chi \in characters G. \chi x = 1$ by metis with χ show False unfolding characters-def by auto qed thus ?thesis using inj-aux hom is-group G'' is-group by (subst inj-on-one-iff) auto qed **lemma** double-dual-iso-eq-iff [simp]:

 $x \in carrier \ G \Longrightarrow y \in carrier \ G \Longrightarrow double-dual-iso \ x = double-dual-iso \ y \longleftrightarrow x = y$

by (*auto dest: inj-onD*[*OF inj-double-dual-iso*])

theorem double-dual-iso: double-dual-iso \in iso G (Characters (Characters G)) **proof** (*rule isoI*) interpret G': finite-comm-group Characters G **by** (*rule finite-comm-group-Characters*) **interpret** G'': finite-comm-group Characters (Characters G) by (rule G'.finite-comm-group-Characters) **show** hom: double-dual-iso \in hom G (Characters (Characters G)) by (rule homI) (auto simp: carrier-Characters characters-def) **show** bij-betw double-dual-iso (carrier G) (carrier (Characters (Characters G))) unfolding *bij-betw-def* proof **show** inj-on double-dual-iso (carrier G) by (fact inj-double-dual-iso) \mathbf{next} **show** double-dual-iso ' carrier G = carrier (Characters (Characters G)) **proof** (*rule card-subset-eq*) **show** finite (carrier (Characters (Characters G))) by (fact G''.fin) \mathbf{next} have card (carrier (Characters (Characters G))) = card (carrier G) by (simp add: carrier-Characters G'.card-characters card-characters order-def) also have $\ldots = card$ (double-dual-iso ' carrier G) **by** (*intro card-image* [*symmetric*] *inj-double-dual-iso*) finally show card (double-dual-iso ' carrier G) = card (carrier (Characters (Characters G)))... next **show** double-dual-iso ' carrier $G \subseteq$ carrier (Characters (Characters G)) using hom by (auto simp: hom-def) \mathbf{qed} qed qed

lemma double-dual-is-iso: Characters (Characters G) \cong G by (rule iso-sym) (use double-dual-iso in (auto simp: is-iso-def))

The second orthogonality relation follows from the first one via Pontryagin duality:

theorem sum-characters: assumes $x: x \in carrier \ G$ shows $(\sum \chi \in characters \ G. \chi \ x) = (if \ x = 1 \ then \ of\ -nat \ (order \ G) \ else \ 0)$ proof – interpret G': finite-comm-group Characters Gby (rule finite-comm-group-Characters) interpret x: character Characters G double-dual-iso xusing x by auto

corollary character-orthogonality2:

assumes $x \in carrier \ G \ y \in carrier \ G$ **shows** $(\sum \chi \in characters \ G. \ \chi \ x \ cnj \ (\chi \ y)) = (if \ x = y \ then \ of-nat \ (order \ G) \ else \ 0)$ **proof from** assms **have** $(\sum \chi \in characters \ G. \ \chi \ x \ cnj \ (\chi \ y)) = (\sum \chi \in characters \ G. \ \chi \ (x \otimes inv \ y))$ **by** $(intro \ sum.cong) \ (simp-all \ add: \ character.char-inv \ character.char-mult \ characters-def)$ **also from** assms **have** $\dots = (if \ x \otimes inv \ y = 1 \ then \ of-nat \ (order \ G) \ else \ 0)$ **by** $(intro \ sum-characters) \ auto$ **also from** assms **have** $x \otimes inv \ y = 1 \ \leftrightarrow x = y$ **using** inv-equality[of x \ inv \ y] **by** auto **finally show** ?thesis . **qed**

end

```
no-notation integer-mod-group (\langle Z \rangle)
end
```

2 Dirichlet Characters

```
theory Dirichlet-Characters

imports

Multiplicative-Characters

HOL–Number-Theory.Residues

Dirichlet-Series.Multiplicative-Function

begin
```

Dirichlet characters are essentially just the characters of the multiplicative group of integer residues $\mathbb{ZZ}/n\mathbb{ZZ}$ for some fixed n. For convenience, these residues are usually represented by natural numbers from 0 to n - 1, and we extend the characters to all natural numbers periodically, so that $\chi(k \mod n) = \chi(k)$ holds.

Numbers that are not coprime to n are not in the group and therefore are assigned 0 by all characters.

2.1 The multiplicative group of residues

definition residue-mult-group :: nat \Rightarrow nat monoid where

residue-mult-group $n = (| carrier = totatives n, monoid.mult = (\lambda x y. (x * y) mod n), one = 1 |)$

definition *principal-dchar* :: $nat \Rightarrow nat \Rightarrow complex$ where principal-dchar $n = (\lambda k. \text{ if coprime } k \text{ n then } 1 \text{ else } 0)$ **lemma** principal-dchar-coprime [simp]: coprime $k \ n \Longrightarrow$ principal-dchar $n \ k = 1$ and principal-dchar-not-coprime [simp]: \neg coprime $k \ n \Longrightarrow$ principal-dchar $n \ k =$ 0 **by** (*simp-all add: principal-dchar-def*) **lemma** principal-dchar-1 [simp]: principal-dchar $n \ 1 = 1$ by simp **lemma** principal-dchar-minus1 [simp]: assumes $n > \theta$ shows principal-dchar $n (n - Suc \ \theta) = 1$ **proof** (cases n = 1) case False with assms have n > 1 by linarith thus *?thesis* using *coprime-diff-one-left-nat*[of n] by (intro principal-dchar-coprime) auto qed auto **lemma** mod-in-totatives: $n > 1 \implies a \mod n \in totatives n \iff coprime \ a \ n$ by (auto simp: totatives-def mod-greater-zero-iff-not-dvd dest: coprime-common-divisor-nat) **bundle** *dcharacter-syntax* begin **notation** principal-dchar $(\langle \chi_0 \rangle)$ end locale residues-nat = fixes n :: nat (structure) and Gassumes n: n > 1defines $G \equiv residue$ -mult-group n begin **lemma** order [simp]: order G = totient nby (simp add: order-def G-def totient-def residue-mult-group-def) **lemma** totatives-mod [simp]: $x \in$ totatives $n \Longrightarrow x \mod n = x$ using n by (intro mod-less) (auto simp: totatives-def intro!: order.not-eq-order-implies-strict) **lemma** principal-dchar-minus1 [simp]: principal-dchar n $(n - Suc \ 0) = 1$ using principal-dchar-minus1 [of n] n by simp sublocale finite-comm-group G proof fix x y assume $xy: x \in carrier G y \in carrier G$

hence coprime (x * y) n

by (*auto simp*: *G-def residue-mult-group-def totatives-def*) with xy and n show $x \otimes_G y \in carrier G$ using coprime-common-divisor-nat[of x * y n] by (auto simp: G-def residue-mult-group-def totatives-def mod-greater-zero-iff-not-dvd le-Suc-eq simp del: coprime-mult-left-iff) \mathbf{next} fix $x \ y \ z$ assume $xyz: x \in carrier \ G \ y \in carrier \ G \ z \in carrier \ G$ thus $x \otimes_G y \otimes_G z = x \otimes_G (y \otimes_G z)$ **by** (*auto simp: G-def residue-mult-group-def mult-ac mod-mult-right-eq*) \mathbf{next} fix x assume $x \in carrier G$ with n have x < n by (auto simp: G-def residue-mult-group-def totatives-def introl: order.not-eq-order-implies-strict) thus $\mathbf{1}_G \otimes_G x = x$ and $x \otimes_G \mathbf{1}_G = x$ **by** (*simp-all add: G-def residue-mult-group-def*) next have $x \in Units \ G$ if $x \in carrier \ G$ for x unfolding Units-def proof safe from that have x > 0 coprime x n**by** (*auto simp*: *G*-def residue-mult-group-def totatives-def) from (coprime x n) and n obtain y where y: $y < n [x * y = 1] \pmod{n}$ **by** (subst (asm) coprime-iff-invertible'-nat) auto hence $x * y \mod n = 1$ using n by (simp add: cong-def mult-ac) moreover from y have coprime y n**by** (subst coprime-iff-invertible-nat) (auto simp: mult.commute) ultimately show $\exists a \in carrier G. a \otimes_G x = \mathbf{1}_G \land x \otimes_G a = \mathbf{1}_G$ using y by (intro bexI[of - y]) (auto simp: G-def residue-mult-group-def totatives-def mult.commute introl: Nat.gr0I)qed fact+thus carrier $G \subseteq Units G$.. **qed** (insert n, auto simp: G-def residue-mult-group-def mult-ac)

2.2 Definition of Dirichlet characters

The following two functions make the connection between Dirichlet characters and the multiplicative characters of the residue group.

definition $c2dc :: (nat \Rightarrow complex) \Rightarrow (nat \Rightarrow complex)$ where $c2dc \ \chi = (\lambda x. \ \chi \ (x \ mod \ n))$ definition $dc2c :: (nat \Rightarrow complex) \Rightarrow (nat \Rightarrow complex)$ where $dc2c \ \chi = (\lambda x. \ if \ x < n \ then \ \chi \ x \ else \ 0)$ lemma $dc2c-c2dc \ [simp]:$ assumes character $G \ \chi$ shows $dc2c \ (c2dc \ \chi) = \chi$ proof interpret character $G \ \chi$ by fact show ?thesis

```
using n by (auto simp: fun-eq-iff dc2c-def c2dc-def char-eq-0-iff G-def
residue-mult-group-def totatives-def)
```

 \mathbf{qed}

 \mathbf{end}

```
locale dcharacter = residues-nat +
 fixes \chi :: nat \Rightarrow complex
 assumes mult-aux: a \in totatives \ n \Longrightarrow b \in totatives \ n \Longrightarrow \chi \ (a * b) = \chi \ a * \chi
b
 assumes eq-zero: \neg coprime \ a \ n \Longrightarrow \chi \ a = 0
 assumes periodic: \chi (a + n) = \chi a
 assumes one-not-zero: \chi \ 1 \neq 0
begin
lemma zero-eq-0 [simp]: \chi \ \theta = \theta
 using n by (intro eq-zero) auto
lemma Suc-0 [simp]: \chi (Suc 0) = 1
 using n mult-aux[of 1 1] one-not-zero by (simp add: totatives-def)
lemma periodic-mult: \chi (a + m * n) = \chi a
proof (induction m)
 case (Suc m)
 have a + Suc \ m * n = a + m * n + n by simp
 also have \chi \ldots = \chi (a + m * n) by (rule periodic)
 also have \ldots = \chi \ a by (rule Suc.IH)
 finally show ?case .
\mathbf{qed} \ simp-all
lemma minus-one-periodic [simp]:
 assumes k > 0
 shows \chi (k * n - 1) = \chi (n - 1)
proof -
 have k * n - 1 = n - 1 + (k - 1) * n
   using assms n by (simp add: algebra-simps)
 also have \chi \ldots = \chi (n - 1)
   by (rule periodic-mult)
 finally show ?thesis .
qed
lemma cong:
 \textbf{assumes} \ [a = b] \ (mod \ n)
 shows \chi a = \chi b
proof –
 from assms obtain k1 \ k2 where *: b + k1 \ * n = a + k2 \ * n
   by (subst (asm) cong-iff-lin-nat) auto
 have \chi \ a = \chi \ (a + k2 * n) by (rule periodic-mult [symmetric])
```

```
also note * [symmetric]
 also have \chi (b + k1 * n) = \chi b by (rule periodic-mult)
 finally show ?thesis .
qed
lemma mod [simp]: \chi (a mod n) = \chi a
 by (rule cong) (simp-all add: cong-def)
lemma mult [simp]: \chi (a * b) = \chi a * \chi b
proof (cases coprime a n \land coprime \ b \ n)
 case True
 hence a mod n \in totatives \ n \ b \ mod \ n \in totatives \ n
  using n by (auto simp: totatives-def mod-greater-zero-iff-not-dvd coprime-absorb-right)
 hence \chi ((a mod n) * (b mod n)) = \chi (a mod n) * \chi (b mod n)
   by (rule mult-aux)
 also have \chi ((a mod n) * (b mod n)) = \chi (a * b)
   by (rule cong) (auto simp: cong-def mod-mult-eq)
 finally show ?thesis by simp
\mathbf{next}
 case False
 hence \neg coprime (a * b) n by simp
 with False show ?thesis by (auto simp: eq-zero)
qed
sublocale mult: completely-multiplicative-function \chi
 by standard auto
lemma eq-zero-iff: \chi x = 0 \leftrightarrow \neg coprime x n
proof safe
 assume \chi x = 0 and coprime x n
 from cong-solve-coprime-nat [OF this(2)]
   obtain y where [x * y = Suc \ 0] \pmod{n} by blast
 hence \chi (x * y) = \chi (Suc \theta) by (rule cong)
 with \langle \chi | x = 0 \rangle show False by simp
qed (auto simp: eq-zero)
lemma minus-one': \chi (n - 1) \in \{-1, 1\}
proof –
 define n' where n' = n - 2
 have n: n = Suc (Suc n') using n by (simp add: n'-def)
 have (n - 1) \hat{2} = 1 + (n - 2) * n
   by (simp add: power2-eq-square algebra-simps n)
 also have \chi \ldots = 1
   by (subst periodic-mult) auto
 also have \chi ((n-1) \hat{2}) = \chi (n-1) \hat{2}
   by (rule mult.power)
 finally show ?thesis
   by (subst (asm) power2-eq-1-iff) auto
qed
```

lemma c2dc-dc2c [simp]: c2dc (dc2c χ) = χ using n by (auto simp: c2dc-def dc2c-def fun-eq-iff intro!: cong simp: cong-def) **lemma** character-dc2c: character G (dc2c χ) by standard (insert n, auto simp: G-def residue-mult-group-def dc2c-def totatives-def *intro*!: *eq-zero*) sublocale dc2c: character G dc2c χ by (fact character-dc2c) **lemma** dcharacter-inv-character [intro]: dcharacter n (inv-character χ) by standard (auto simp: inv-character-def eq-zero periodic) **lemma** norm: norm $(\chi k) = (if \ coprime \ k \ n \ then \ 1 \ else \ 0)$ proof have $\chi k = \chi (k \mod n)$ by (intro cong) (auto simp: cong-def) also from n have ... = $dc2c \ \chi \ (k \ mod \ n)$ by $(simp \ add: \ dc2c \ def)$ also from *n* have norm $\ldots = (if \ coprime \ k \ n \ then \ 1 \ else \ 0)$ by (subst dc2c.norm-char) (auto simp: G-def residue-mult-group-def mod-in-totatives) finally show ?thesis . qed **lemma** norm-le-1: norm $(\chi k) \leq 1$ by (subst norm) auto end definition dcharacters :: $nat \Rightarrow (nat \Rightarrow complex)$ set where dcharacters $n = \{\chi. \ dcharacter \ n \ \chi\}$ **context** residues-nat begin **lemma** character-dc2c: dcharacter n $\chi \Longrightarrow$ character G (dc2c χ) **using** dcharacter.character.dc2c[of $n \chi$] **by** (simp add: G-def) **lemma** *dcharacter-c2dc*: assumes character $G \chi$ **shows** dcharacter n ($c2dc \chi$) proof – **interpret** character $G \ \chi$ by fact **show** ?thesis proof fix x assume $\neg coprime \ x \ n$ thus $c2dc \ \chi \ x = 0$ by (auto simp: c2dc-def char-eq-0-iff G-def residue-mult-group-def totatives-def) qed (insert char-mult char-one n, auto simp: c2dc-def G-def residue-mult-group-def simp del: char-mult char-one) qed **lemma** principal-dchar-altdef: principal-dchar n = c2dc (principal-char G) using n by (auto simp: c2dc-def principal-dchar-def principal-char-def G-def residue-mult-group-def fun-eq-iff mod-in-totatives) sublocale principal: dcharacter $n \ G$ principal-dchar nby (simp add: principal-dchar-altdef dcharacter- $c2dc \mid rule \ G-def)+$ lemma c2dc-principal [simp]: c2dc (principal-char G) = principal-dchar n **by** (*simp add: principal-dchar-altdef*) **lemma** dc2c-principal [simp]: dc2c (principal-dchar n) = principal-char G proof have dc2c (c2dc (principal-char G)) = dc2c (principal-dchar n) by (subst c2dc-principal) (rule refl) thus ?thesis by (subst (asm) dc2c-c2dc) simp-all qed

lemma finite-dcharacters [intro]: finite (dcharacters n) using bij-betw-finite [OF bij-betw-dcharacters-characters] by auto

lemma card-dcharacters [simp]: card (dcharacters n) = totient nusing bij-betw-same-card [OF bij-betw-dcharacters-characters] card-characters by simp

end

```
lemma inv-character-eq-principal-dchar-iff [simp]:
inv-character \chi = principal-dchar n \leftrightarrow \chi = principal-dchar n
by (auto simp add: fun-eq-iff inv-character-def principal-dchar-def)
```

2.3 Sums of Dirichlet characters

lemma (in *dcharacter*) sum-dcharacter-totatives: $(\sum x \in totatives n, \chi x) = (if \chi = principal-dchar n then of-nat (totient n) else$ θ) proof from *n* have $(\sum x \in totatives n, \chi x) = (\sum x \in carrier G, dc2c \chi x)$ by (intro sum.cong) (auto simp: totatives-def dc2c-def G-def residue-mult-group-def) also have ... = (if dc2c χ = principal-char G then of-nat (order G) else 0) by (rule dc2c.sum-character) also have $dc2c \ \chi = principal-char \ G \longleftrightarrow \chi = principal-dchar \ n$ **by** (*metis c2dc-dc2c dc2c-principal principal-dchar-altdef*) finally show ?thesis by simp qed **lemma** (in dcharacter) sum-dcharacter-block: $(\sum x < n, \chi x) = (if \chi = principal dchar n then of nat (totient n) else 0)$ proof from *n* have $(\sum x < n. \chi x) = (\sum x \in totatives n. \chi x)$ **by** (*intro sum.mono-neutral-right*) (auto simp: totatives-def eq-zero-iff intro!: Nat.gr0I order.not-eq-order-implies-strict) also have ... = (if χ = principal-dchar n then of-nat (totient n) else 0) by (rule sum-dcharacter-totatives) finally show ?thesis . qed **lemma** (in *dcharacter*) sum-dcharacter-block': $sum \chi \{Suc \ 0..n\} = (if \ \chi = principal-dchar \ n \ then \ of-nat \ (totient \ n) \ else \ 0)$ proof let $?f = \lambda k$. if k = n then 0 else k and $?g = \lambda k$. if k = 0 then n else k have sum χ {1..n} = sum χ {..<n} using n by (intro sum.reindex-bij-witness [where j = ?f and i = ?q]) (auto simp: eq-zero-iff) **thus** ?thesis **by** (simp add: sum-dcharacter-block) qed **lemma** (in dcharacter) sum-lessThan-dcharacter: assumes $\chi \neq principal$ -dchar n shows $(\sum x < m. \chi x) = (\sum x < m \mod n. \chi x)$ **proof** (*induction m rule: less-induct*) case (less m) show ?case **proof** (cases m < n) case True thus ?thesis by simp \mathbf{next} case False hence $\{..< m\} = \{..< m\} \cup \{n..< m\}$ by *auto* also have $(\sum x \in \ldots \chi x) = (\sum x < n. \chi x) + (\sum x \in \{n.. < m\}, \chi x)$ by (intro sum.union-disjoint) auto

also from assms have $(\sum x < n, \chi x) = 0$ **by** (subst sum-dcharacter-block) simp-all also from False have $(\sum x \in \{n ... < m\}, \chi x) = (\sum x \in \{... < m - n\}, \chi (x + n))$ by (intro sum.reindex-bij-witness[of - λx . $x + n \lambda x$. x - n]) (auto simp: *periodic*) also have $\dots = (\sum x \in \{..< m - n\}, \chi x)$ by (simp add: periodic) also have $\dots = (\sum x < (m - n) \mod n, \chi x)$ using False and n by (intro less.IH) auto also from *False* and *n* have $(m - n) \mod n = m \mod n$ **by** (*simp add: le-mod-geq*) finally show ?thesis by simp qed qed **lemma** (in *dcharacter*) *sum-dcharacter-lessThan-le*: assumes $\chi \neq principal$ -dchar n **shows** norm $(\sum x < m. \chi x) \le totient n$ proof have $(\sum x < m. \chi x) = (\sum x < m \mod n. \chi x)$ by (rule sum-less Than-dcharacter) fact also have $\ldots = (\sum x \mid x < m \mod n \land coprime \ x \ n. \ \chi \ x)$ by (intro sum.mono-neutral-right) (auto simp: eq-zero-iff) also have norm $\ldots \leq (\sum x \mid x < m \mod n \land coprime x n. 1)$ **by** (*rule sum-norm-le*) (*auto simp: norm*) also have $\ldots = card \{x. x < m \mod n \land coprime x n\}$ by simp also have $\ldots \leq card$ (totatives n) unfolding of-nat-le-iff **proof** (*intro* card-mono subsetI) fix x assume $x: x \in \{x. x < m \mod n \land coprime x n\}$ hence $x < m \mod n$ by simp also have $\ldots < n$ using n by simpfinally show $x \in totatives \ n \text{ using } x$ **by** (*auto simp: totatives-def intro*!: *Nat.gr0I*) qed auto also have \ldots = totient n by (simp add: totient-def) finally show ?thesis . qed **lemma** (in *dcharacter*) *sum-dcharacter-atMost-le*: assumes $\chi \neq principal$ -dchar n **shows** norm $(\sum x \le m, \chi x) \le totient n$ using sum-dcharacter-less Than-le[OF assms, of Suc m] by (subst (asm) less Than-Suc-atMost) **lemma** (in residues-nat) sum-dcharacters: $(\sum \chi \in dcharacters \ n. \ \chi \ x) = (if \ [x = 1] \ (mod \ n) \ then \ of-nat \ (totient \ n) \ else \ 0)$ **proof** (cases coprime x n) case True with *n* have *x*: *x* mod $n \in$ totatives *n* by (auto simp: mod-in-totatives) have $(\sum \chi \in dcharacters \ n. \ \chi \ x) = (\sum \chi \in characters \ G. \ c2dc \ \chi \ x)$ by (rule sum.reindex-bij-betw [OF bij-betw-characters-dcharacters, symmetric])

also from x have $\ldots = (\sum \chi \in characters \ G. \ \chi \ (x \ mod \ n))$ by (simp add: c2dc-def) **also from** x have $\ldots = (if x \mod n = 1 \text{ then order } G \text{ else } 0)$ by (subst sum-characters) (unfold G-def residue-mult-group-def, auto) also from *n* have $x \mod n = 1 \iff [x = 1] \pmod{n}$ **by** (*simp add: conq-def*) finally show ?thesis by simp \mathbf{next} case False have $x \mod n \neq 1$ proof assume $*: x \mod n = 1$ have $gcd (x \mod n) n = 1$ by (subst *) simpalso have $gcd (x \mod n) n = gcd x n$ by (subst gcd.commute) (simp only: gcd-red-nat [symmetric]) finally show False using $\langle \neg coprime \ x \ n \rangle$ unfolding coprime-iff-qcd-eq-1 by contradictionqed from False have $(\sum \chi \in dcharacters \ n. \ \chi \ x) = 0$ by (intro sum.neutral) (auto simp: dcharacters-def dcharacter.eq-zero) with $\langle x \mod n \neq 1 \rangle$ and n show ?thesis by (simp add: cong-def) qed **lemma** (in dcharacter) even-dcharacter-linear-sum-eq-0 [simp]: assumes $\chi \neq principal$ -dchar n and $\chi (n - 1) = 1$ shows $(\sum k=Suc \ 0..< n. \ of-nat \ k * \chi \ k) = 0$ proof have $(\sum k=1..< n. \text{ of-nat } k * \chi k) = (\sum k=1..< n. (\text{of-nat } n - \text{ of-nat } k) * \chi (n)$ (-k))by (intro sum.reindex-bij-witness [where $i = \lambda k$. n - k and $j = \lambda k$. n - k]) (auto simp: of-nat-diff) also have ... = $n * (\sum k=1..< n. \chi (n-k)) - (\sum k=1..< n. k * \chi (n-k))$ **by** (simp add: algebra-simps sum-subtractf sum-distrib-left) also have $(\sum k=1..< n. \chi (n-k)) = (\sum k=1..< n. \chi k)$ by (intro sum.reindex-bij-witness [where $i = \lambda k$. n - k and $j = \lambda k$. n - k]) autoalso have $\ldots = (\sum k < n. \chi k)$ by (intro sum.mono-neutral-left) (auto simp: Suc-le-eq) also have $\ldots = 0$ using assms by (simp add: sum-dcharacter-block) **also have** $(\sum k=1..< n. \text{ of-nat } k * \chi (n-k)) = (\sum k=1..< n. k * \chi k)$ **proof** (*intro sum.cong refl*) fix k assume $k: k \in \{1 ... < n\}$ have of-nat $k * \chi k = of$ -nat $k * \chi ((n - 1) * k)$ using assms by (subst mult) simp-all also have (n - 1) * k = n - k + (k - 1) * nusing k by (simp add: algebra-simps) also have $\chi \ldots = \chi (n - k)$ by (rule periodic-mult) finally show of-nat $k * \chi (n - k) = of$ -nat $k * \chi k ...$

```
qed
finally show ?thesis by simp
qed
```

end

3 Dirichlet *L*-functions

```
theory Dirichlet-L-Functions
imports
Dirichlet-Characters
HOL-Library.Landau-Symbols
Zeta-Function.Zeta-Function
begin
```

We can now define the Dirichlet *L*-functions. These are essentially the functions in the complex plane that the Dirichlet series $\sum_{k=1}^{\infty} \chi(k) k^{-s}$ converge to, for some fixed Dirichlet character χ .

First of all, we need to take care of a syntactical problem: The notation for vectors uses χ as syntax, which causes some annoyance to us, so we disable it locally.

3.1 Definition and basic properties

We now define Dirichlet L functions as a finite linear combination of Hurwitz ζ functions. This has the advantage that we directly get the analytic continuation over the full domain and only need to prove that the series really converges to this definition whenever it does converge, which is not hard to do.

definition Dirichlet-L :: nat \Rightarrow (nat \Rightarrow complex) \Rightarrow complex \Rightarrow complex where Dirichlet-L m χ s =

 $\begin{array}{l} (if \ s = 1 \ then \\ if \ \chi = principal dchar \ m \ then \ 0 \ else \ eval fds \ (fds \ \chi) \ 1 \\ else \\ of -nat \ m \ powr \ - \ s \ * \ (\sum k = 1..m. \ \chi \ k \ * \ hurwitz - zeta \ (real \ k \ / \ real \ m) \ s)) \end{array}$

lemma Dirichlet-L-conv-hurwitz-zeta-nonprincipal:

assumes $s \neq 1$ shows Dirichlet-L n χ s = of-nat n powr $-s * (\sum k = 1..n. \chi k * hurwitz-zeta (real k / real n) s)$ using assms by (simp add: Dirichlet-L-def)

Analyticity everywhere except 1 is trivial by the above definition, since the Hurwitz ζ function is analytic everywhere except 1. For L functions of non principal characters, we will have to show the analyticity at 1 separately later.

lemma holomorphic-Dirichlet-L-weak: assumes m > 0 1 $\notin A$ shows Dirichlet-L $m \chi$ holomorphic-on A proof – have $(\lambda s. of-nat \ m \ powr - s * (\sum k = 1..m. \chi \ k * hurwitz-zeta (real k / real$ m) s))holomorphic-on Ausing assms unfolding Dirichlet-L-def by (intro holomorphic-intros) auto $also have ?this <math>\leftrightarrow$?thesis using assms by (intro holomorphic-cong refl) (auto simp: Dirichlet-L-def) finally show ?thesis . qed

context dcharacter begin

For a real value greater than 1, the formal Dirichlet series of an L function for some character χ converges to the L function.

lemma

fixes s :: complex assumes s: $Re \ s > 1$ abs-summable-Dirichlet-L: summable (λn . norm ($\chi n * of$ -nat n powr shows -s))*summable-Dirichlet-L*: summable ($\lambda n. \chi n * of-nat n powr -s$) and *sums-Dirichlet-L*: $(\lambda n. \ \chi \ n * n \ powr \ -s)$ sums Dirichlet-L $n \ \chi \ s$ and Dirichlet-L-conv-eval-fds-weak: Dirichlet-L n χ s = eval-fds (fds χ) s and proof define L where $L = (\sum n. \chi n * of\text{-}nat n powr - s)$ **show** summable $(\lambda n. norm (\chi n * of-nat n powr -s))$ by (subst summable-Suc-iff [symmetric], rule summable-comparison-test [OF - summable-zeta-real[of Re s]]) (insert s norm, auto intro!: exI[of - 0] simp: norm-mult norm-powr-real-powr) **thus** summable: summable $(\lambda n. \chi n * of-nat n powr -s)$ by (rule summable-norm-cancel) hence $(\lambda n. \ \chi \ n * of-nat \ n \ powr \ -s)$ sums L by $(simp \ add: L-def \ sums-iff)$ from this have $(\lambda m. \sum k = m * n.. < m * n + n. \chi k * of-nat k powr - s)$ sums L by (rule sums-group) (use n in auto) also have $(\lambda m. \sum k = m * n.. < m * n + n. \chi k * of-nat k powr - s) = (\lambda m. of-nat n powr - s * (\sum k = 1..n. \chi k * (of-nat m + of-nat k / nat m + of-nat m + of-n$ of-nat n) powr (-s)**proof** (*rule ext*, *goal-cases*) case (1 m)have $(\sum_{n=1}^{\infty} k = m * n ... < m * n + n. \chi k * of-nat k powr - s) =$ $(\sum k=0..< n. \ \chi \ (k + m * n) * of-nat \ (m * n + k) \ powr - s)$ by (intro sum.reindex-bij-witness[of - λk . $k + m * n \lambda k$. k - m * n]) auto also have ... = $(\sum k=0..<n, \chi k * of-nat (m * n + k) powr - s)$

by (simp add: periodic-mult) also have $\ldots = (\sum k = 0 \dots < n. \chi k * (of-nat m + of-nat k / of-nat n) powr$ s * of-nat n powr -s) **proof** (*intro sum.cong refl, goal-cases*) case (1 k)have of-nat (m * n + k) = (of-nat m + of-nat k / of-nat n :: complex) *of-nat nusing n by (simp add: divide-simps del: div-mult-self1 div-mult-self2 div-mult-self3 div-mult-self4) also have ... powr $-s = (of-nat \ m + of-nat \ k \ / of-nat \ n)$ powr -s * of-natn powr - sby (rule powr-times-real) auto finally show ?case by simp qed also have $\ldots = of$ -nat n powr $-s * (\sum k=0 ... < n. \chi k * (of$ -nat m + of-nat k/ of-nat n) powr -s) **by** (*subst sum-distrib-left*) (*simp-all add: mult-ac*) also have $(\sum k = 0..< n. \ \chi \ k * (of-nat \ m + of-nat \ k \ / of-nat \ n) \ powr - s) = (\sum k = 1..< n. \ \chi \ k * (of-nat \ m + of-nat \ k \ / of-nat \ n) \ powr - s)$ $\mathbf{by} \ (intro \ sum.mono-neutral-right) \ (auto \ simp: \ Suc-le-eq)$ also have $\ldots = (\sum k = 1 \dots \chi k * (of-nat m + of-nat k / of-nat n) powr$ s)using periodic-mult[of 0 1] by (intro sum.mono-neutral-left) auto finally show ?case . qed finally have ... sums L. moreover have $(\lambda m. of-nat \ n \ powr - s * (\sum k=1..n. \ \chi \ k * (of-nat \ m + of-real)))$ $(of-nat \ k \ / \ of-nat \ n)) \ powr \ - \ s)) \ sums$ (of-nat n powr - $s * (\sum k=1..n. \chi k * hurwitz-zeta (of-nat k /$ of-nat n) s)) using s by (intro sums-sum sums-mult sums-hurwitz-zeta) auto ultimately have $L = \ldots$ **by** (*simp add: sums-iff*) also have ... = Dirichlet-L n χ s using assms by (auto simp: Dirichlet-L-def) finally have Dirichlet-L $n \chi s = (\sum n. \chi n * of-nat n powr -s)$ by (simp add: L-def) with summable show $(\lambda n. \ \chi \ n * n \ powr \ -s)$ sums Dirichlet-L $n \ \chi \ s$ by (simp add: sums-iff L-def) **thus** Dirichlet-L $n \chi s = eval-fds$ (fds χ) s by (simp add: eval-fds-def sums-iff powr-minus field-simps fds-nth-fds') \mathbf{qed} **lemma** fds-abs-converges-weak: Re $s > 1 \implies$ fds-abs-converges (fds χ) s using abs-summable-Dirichlet-L[of s] by (simp add: fds-abs-converges-def powr-minus divide-simps fds-nth-fds')

lemma abs-conv-abscissa-weak: abs-conv-abscissa (fds χ) ≤ 1 **proof** (rule abs-conv-abscissa-leI, goal-cases) **case** (1 c) thus ?case

by (*intro* exI[of - of-real c] conjI fds-abs-converges-weak) auto **qed**

Dirichlet L functions have the Euler product expansion

$$L(\chi, s) = \prod_{p} \left(1 - \frac{\chi(p)}{p^{-s}} \right)$$

for all s with $\Re(s) > 1$.

lemma

else 1)

(**is** ?th2)

proof -

have mult: completely-multiplicative-function (fds-nth (fds χ)) using mult.completely-multiplicative-function-axioms by (simp add: fds-nth-fds')

have conv: fds-abs-converges (fds χ) s

using abs-summable-Dirichlet-L[OF s]

by (*simp add: fds-abs-converges-def fds-nth-fds' powr-minus divide-simps*)

have $(\lambda n. \prod p \le n. if prime p then inverse (1 - \chi p / nat-power p s) else 1)$ $\longrightarrow eval-fds (fds \chi) s$

using fds-euler-product-LIMSEQ' [OF mult conv] **by** (simp add: fds-nth-fds' cong: if-cong)

also have eval-fds (fds χ) $s = Dirichlet-L n \chi s$ using sums-Dirichlet-L[OF s] unfolding eval-fds-def

by (simp add: sums-iff fds-nth-fds' powr-minus divide-simps)

finally show ?th1.

```
\label{eq:from fds-abs-convergent-euler-product' [OF mult conv] show ?th2
```

by (*simp add: fds-nth-fds cong: if-cong*)

```
\mathbf{qed}
```

also have eval-fds (fds χ) $s = Dirichlet-L n \chi s$ using Dirichlet-L-conv-eval-fds-weak[OF assms] by simp **also have** $\neg(\exists p. prime p \land fds\text{-}nth (fds \chi) p = nat\text{-}power p s)$ **proof** safe **fix** p :: nat assume p: prime p fds-nth (fds χ) p = nat-power p s from p have real 1 < real p by (subst of-nat-less-iff) (auto simp: prime-gt-Suc-0-nat) also have \ldots = real p powr 1 by simp also from p and assms have real p powr $1 \leq real p$ powr Re sby (intro powr-mono) (auto simp: real-of-nat-ge-one-iff prime-ge-Suc-0-nat) also have $\ldots = norm (nat-power p s)$ by (simp add: norm-nat-power norm-powr-real-powr)also have *nat-power* $p \ s = fds$ -*nth* (*fds* χ) p using p by *simp* also have norm $\ldots \leq 1$ by (auto simp: fds-nth-fds' norm) finally show False by simp qed finally show ?thesis . qed lemma sum-dcharacter-antimono-bound: fixes $x0 \ a \ b :: real$ and $ff' :: real \Rightarrow real$ assumes nonprincipal: $\chi \neq \chi_0$ **assumes** $x\theta$: $x\theta \ge \theta$ and ab: $x\theta \le a \ a < b$ assumes $f': \bigwedge x. \ x \ge x0 \implies (f \text{ has-field-derivative } f' x) \ (at x)$ assumes f-nonneg: $\bigwedge x. \ x \ge x0 \implies f \ x \ge 0$ assumes f'-nonpos: $\bigwedge x. \ x \ge x0 \implies f' \ x \le 0$ shows norm $(\sum n \in real - (a < ... b), \chi n * (f (real n))) \le 2 * real (totient n)$ *faproof **note** deriv = has-field-derivative-at-within [OF f'] let ?A = sum-upto χ have cont: continuous-on $\{a..b\}$ f by (rule DERIV-continuous-on[OF deriv]) (use ab in auto) have I': $(f' has-integral (f b - f a)) \{a..b\}$ using ab deriv by (intro fundamental-theorem-of-calculus) (auto simp: has-real-derivative-iff-has-vector-derivative [symmetric]) define I where $I = integral \{a...b\} (\lambda t. ?A t * of-real (f' t))$ define C where C = real (totient n) have C-nonneg: $C \ge 0$ by (simp add: C-def) have C: norm $(?A x) \leq C$ for x proof have ?A $x = (\sum k \le nat \lfloor x \rfloor, \chi k)$ unfolding sum-upto-altdef **by** (*intro sum.mono-neutral-left*) *auto* also have norm $\ldots \leq C$ unfolding C-def using nonprincipal **by** (*rule sum-dcharacter-atMost-le*) finally show ?thesis . qed have I: $((\lambda t. ?A t * f' t) has-integral ?A b * f b - ?A a * f a -$

 $(\sum n \in real - (\{a < ...b\}, \chi n * f (real n))) \{a...b\}$ using $ab \ x0 \ cont f'$

by (intro partial-summation-strong of {}] has-vector-derivative-of-real) auto hence $(\sum n \in real - (a < ... b), \chi n * f (real n)) = ?A b * f b - ?A a * f a - I$ **by** (*simp add: has-integral-iff I-def*) also have norm $\ldots \leq norm$ (?A b) * norm (f b) + norm (?A a) * norm (f a) + norm Iby (rule order.trans[OF norm-triangle-ineq4] add-mono)+ (simp-all add: norm-mult) also have norm $I \leq integral \{a..b\}$ ($\lambda t. of-real (-C) * of-real (f' t)$) unfolding I-def using I I' f'-nonpos ab C by (intro integral-norm-bound-integral integrable-on-cmult-left) (simp-all add: has-integral-iff norm-mult mult-right-mono-neg) **also have** ... = -(C * (f b - f a))using integral-linear [OF - bounded-linear-of-real, of $f' \{a..b\}$] I' **by** (simp add: has-integral-iff o-def) also have $\ldots = C * (f a - f b)$ by (simp add: algebra-simps) also have norm (sum-up to χ b) $\leq C$ by (rule C) also have norm (sum-up to χ a) < C by (rule C) **also have** C * norm (f b) + C * norm (f a) + C * (f a - f b) = 2 * C * f a**using** *f*-nonneg[of a] *f*-nonneg[of b] ab **by** (simp add: algebra-simps) finally show ?thesis by (simp add: mult-right-mono C-def) qed **lemma** summable-dcharacter-antimono: **fixes** $x0 \ a \ b :: real$ and $ff' :: real \Rightarrow real$ assumes nonprincipal: $\chi \neq \chi_0$ assumes $f': \bigwedge x. \ x \ge x0 \implies (f \text{ has-field-derivative } f' x) \ (at \ x)$ assumes f-nonneg: $\bigwedge x. \ x \ge x0 \implies f \ x \ge 0$ assumes f'-nonpos: $\bigwedge x. \ x \ge x0 \implies f' \ x \le 0$ assumes $lim: (f \longrightarrow 0) \ at$ -top shows summable $(\lambda n. \chi n * f n)$ **proof** (rule summable-bounded-partials [where $?g = \lambda x$. 2 * real (totient n) * f x, goal-cases) case 1 **from** eventually-ge-at-top[of nat $\lceil x0 \rceil$] **show** ?case **proof** eventually-elim case (elim x) show ?case **proof** (*safe*, *goal-cases*) case $(1 \ a \ b)$ with elim have $*: max \ 0 \ x0 \ge 0 \ max \ 0 \ x0 \le a \ real \ a < real \ b$ by (simp-all add: nat-le-iff ceiling-le-iff) have $(\sum n \in \{a < ...b\}, \chi n * complex-of-real (f (real n))) =$ $(\sum n \in real - ` \{real \ a < ... real \ b \}. \ \chi \ n \ * \ complex \text{-of-real} \ (f \ (real \ n)))$ by (intro sum.cong refl) auto also have norm $\ldots \leq 2 * real$ (totient n) * f ausing nonprincipal * f' f-nonneg f'-nonpos by (rule sum-dcharacter-antimono-bound) simp-all finally show ?case . qed qed

qed (auto intro!: tendsto-mult-right-zero filterlim-compose[OF lim] filterlim-real-sequentially)

lemma conv-abscissa-le-0: fixes s :: realassumes nonprincipal: $\chi \neq \chi_0$ **shows** conv-abscissa (fds χ) ≤ 0 **proof** (rule conv-abscissa-leI) fix s assume s: 0 < ereal shave summable $(\lambda n. \ \chi \ n * of-real \ (n \ powr \ -s))$ **proof** (rule summable-dcharacter-antimono[of 1]) fix x :: real assume $x \ge 1$ thus $((\lambda x. x powr - s) has field - derivative (-s * x powr (-s-1)))$ (at x) **by** (*auto intro*!: *derivative-eq-intros*) **qed** (insert s assms, auto introl: tendsto-neg-powr filterlim-ident) **thus** $\exists s'::complex. s' \cdot 1 = s \land fds\text{-}converges (fds \chi) s' using s$ by (intro exI[of - of - real s]) (auto simp: fds-converges-def powr-minus divide-simps powr-of-real [symmetric] fds-nth-fds') \mathbf{qed}

```
lemma summable-Dirichlet-L':

assumes nonprincipal: \chi \neq \chi_0

assumes s: Re s > 0

shows summable (\lambda n. \chi n * of-nat n powr -s)

proof –

from assms have fds-converges (fds \chi) s

by (intro fds-converges le-less-trans[OF conv-abscissa-le-0]) auto

thus ?thesis by (simp add: fds-converges-def powr-minus divide-simps fds-nth-fds')

qed
```

lemma

assumes $\chi \neq \chi_0$ shows Dirichlet-L-conv-eval-fds: Λs . Re $s > 0 \Longrightarrow$ Dirichlet-L n $\chi s = eval-fds$ $(fds \ \chi) \ s$ and holomorphic-Dirichlet-L: Dirichlet-L n χ holomorphic-on A proof **show** eq: Dirichlet-L n χ s = eval-fds (fds χ) s (is ?f s = ?g s) if Re s > 0 for s **proof** (cases s = 1) case False show ?thesis **proof** (rule analytic-continuation-open[where ?f = ?f and ?g = ?g]) **show** $\{s. Re \ s > 1\} \subseteq \{s. Re \ s > 0\} - \{1\}$ by *auto* show connected ({s. 0 < Re s} - {1}) using aff-dim-halfspace-gt[of 0 1::complex] by (intro connected-punctured-convex convex-halfspace-Re-gt) auto qed (insert that n assms False, auto intro!: convex-halfspace-Re-gt open-halfspace-Re-gt exI[of - 2] holomorphic-intros holomorphic-Dirichlet-L-weak Dirichlet-L-conv-eval-fds-weak le-less-trans[OF conv-abscissa-le-0])

qed (*insert assms*, *simp-all add: Dirichlet-L-def*)

have Dirichlet-L n χ holomorphic-on UNIV **proof** (rule no-isolated-singularity') from *n* show Dirichlet-L *n* χ holomorphic-on (UNIV - {1}) **by** (*intro holomorphic-Dirichlet-L-weak*) *auto* \mathbf{next} fix s :: complex assume $s: s \in \{1\}$ show Dirichlet-L n χ -s \rightarrow Dirichlet-L n χ s **proof** (rule Lim-transform-eventually) from assms have continuous-on {s. Re s > 0} (eval-fds (fds χ)) by (intro holomorphic-fds-eval holomorphic-on-imp-continuous-on) (auto intro: le-less-trans[OF conv-abscissa-le-0]) hence eval-fds (fds χ) $-s \rightarrow$ eval-fds (fds χ) s using s $\mathbf{by} \ (subst \ (asm) \ continuous \text{-} on \text{-} eq\text{-} continuous \text{-} at) \ (auto \ simp: \ open-halfspace-Re-gt)$ *isCont-def*) also have eval-fds (fds χ) $s = Dirichlet-L n \chi s$ using assms s by (simp add: Dirichlet-L-def) finally show eval-fds (fds χ) $-s \rightarrow Dirichlet-L \ n \ \chi \ s$. \mathbf{next} have eventually (λz . $z \in \{z. \text{ Re } z > 0\}$) (nhds s) using s by (intro eventually-nhds-in-open) (auto simp: open-halfspace-Re-gt) hence eventually ($\lambda z. \ z \in \{z. \ Re \ z > 0\}$) (at s) unfolding eventually-at-filter by eventually-elim auto then show eventually (λz . eval-fds (fds χ) $z = Dirichlet-L n \chi z$) (at s) by eventually-elim (auto intro!: eq [symmetric]) qed qed auto thus Dirichlet-L n χ holomorphic-on A by (rule holomorphic-on-subset) auto qed **lemma** *cnj-Dirichlet-L*: cnj (Dirichlet-L n χ s) = Dirichlet-L n (inv-character χ) (cnj s) proof -{ assume *: $\chi \neq \chi_0 \ s = 1$ with summable-Dirichlet-L'[of 1] have $(\lambda n, \chi n / n)$ sums eval-fds (fds χ) 1 by (simp add: eval-fds-def fds-nth-fds' powr-minus sums-iff divide-simps) hence $(\lambda n. inv-character \chi n / n)$ sums cnj (eval-fds (fds $\chi)$ 1) by (subst (asm) sums-cnj [symmetric]) (simp add: inv-character-def) hence eval-fds (fds (inv-character χ)) 1 = cnj (eval-fds (fds χ) 1) by (simp add: eval-fds-def fds-nth-fds' inv-character-def sums-iff) } thus ?thesis by (auto simp add: Dirichlet-L-def cnj-powr eval-inv-character) qed end

lemma holomorphic-Dirichlet-L [holomorphic-intros]: assumes n > 1 $\chi \neq principal$ -dchar $n \wedge$ dcharacter $n \chi \vee \chi = principal$ -dchar $n \wedge 1 \notin A$ shows Dirichlet-L $n \chi$ holomorphic-on A using assms(2)proof assume $\chi = principal-dchar \ n \land 1 \notin A$ with holomorphic-Dirichlet-L-weak [of $n \ A \ principal$ -dchar n] assms(1) show *?thesis* by *auto* **qed** (insert dcharacter.holomorphic-Dirichlet-L[of $n \chi A$], auto) **lemma** *holomorphic-Dirichlet-L'* [*holomorphic-intros*]: assumes n > 1 f holomorphic-on A $\chi \neq principal$ -dchar $n \land dcharacter n \chi \lor \chi = principal$ -dchar $n \land (\forall x \in A.$ $f x \neq 1$) **shows** ($\lambda s.$ Dirichlet-L n χ (f s)) holomorphic-on A using holomorphic-on-compose[OF assms(2) holomorphic-Dirichlet-L[OF assms(1),of χ]] assms by (auto simp: o-def image-iff) **lemma** continuous-on-Dirichlet-L: assumes n > 1 $\chi \neq principal$ -dchar $n \wedge dcharacter n \chi \vee \chi = principal$ -dchar $n\,\wedge\,1\,\notin\,A$ **shows** continuous-on A (Dirichlet-L $n \chi$) using assms by (intro holomorphic-on-imp-continuous-on holomorphic-intros) **lemma** continuous-on-Dirichlet-L' [continuous-intros]: **assumes** continuous-on A f n > 1and $\chi \neq principal$ -dchar $n \wedge dcharacter n \chi \vee \chi = principal$ -dchar $n \wedge$ $(\forall x \in A. f x \neq 1)$ **shows** continuous-on A (λx . Dirichlet-L n χ (f x)) using continuous-on-compose2[OF continuous-on-Dirichlet-L[of $n \chi f \cdot A$] assms(1)] assms by (auto simp: image-iff) **corollary** continuous-Dirichlet-L [continuous-intros]: $n > 1 \Longrightarrow \chi \neq principal-dchar n \land dcharacter n \chi \lor \chi = principal-dchar n \land s$ $\neq 1 \Longrightarrow$ continuous (at s within A) (Dirichlet-L n χ) **by** (*rule continuous-within-subset*[*of* - *UNIV*]) (insert continuous-on-Dirichlet-L[of $n \chi$ (if $\chi = principal-dchar n$ then $-\{1\}$ else UNIV], auto simp: continuous-on-eq-continuous-at open-Compl) **corollary** continuous-Dirichlet-L' [continuous-intros]: $n > 1 \Longrightarrow$ continuous (at s within A) $f \Longrightarrow$ $\chi \neq principal-dchar \ n \ \land \ dcharacter \ n \ \chi \lor \chi = principal-dchar \ n \land f \ s \neq 1$ continuous (at s within A) (λx . Dirichlet-L n χ (f x)) by (rule continuous-within-compose $\Im[OF \text{ continuous-Dirichlet-}L]$) auto

context residues-nat begin

Applying the above to the $L(\chi_0, s)$, the L function of the principal character, we find that it differs from the Riemann ζ function only by multiplication with a constant that depends only on the modulus n. They therefore have the same analytic properties as the ζ function itself.

lemma Dirichlet-L-principal:

fixes s :: complex **shows** Dirichlet-L $n \chi_0 s = (\prod p \mid prime p \land p dvd n. (1 - 1 / p powr s)) *$ zeta s(**is** ?f s = ?g s)**proof** (cases s = 1) case False show ?thesis **proof** (rule analytic-continuation-open [where ?f = ?f and ?g = ?g]) show $\{s. Re \ s > 1\} \subseteq -\{1\}$ by *auto* **show** ?f s = ?g s **if** $s \in \{s. Re \ s > 1\}$ **for** sproof from that have s: $Re \ s > 1$ by simp let $?P = (\prod p \mid prime p \land p \ dvd \ n. \ (1 - 1 \ / \ p \ powr \ s))$ have $(\lambda n, \prod p \leq n, if prime p then inverse (1 - \chi_0 p / nat-power p s) else 1)$ \rightarrow Dirichlet-L n χ_0 s using s by (rule principal.Dirichlet-L-euler-product-LIMSEQ) also have ?this \leftrightarrow $(\lambda n. ?P * (\prod p \le n. if prime p then inverse (1 - 1 /$ of-nat p powr s) else 1)) \longrightarrow Dirichlet-L n χ_0 s (is - = filterlim ?g - -) **proof** (*intro tendsto-cong eventually-mono* [OF eventually-ge-at-top, of n], goal-cases) case (1 m)let $?f = \lambda p$. inverse (1 - 1 / p powr s)have $(\prod p \le m. if prime p then inverse (1 - \chi_0 p / nat-power p s) else 1) =$ $(\prod p \mid p \leq m \land prime p \land coprime p n. ?f p)$ (is - = prod - ?A) by (intro prod.mono-neutral-cong-right) (auto simp: principal-dchar-def) also have $?A = \{p, p \leq m \land prime p\} - \{p, prime p \land p dvd n\}$ (is - ?B - ?C) using n by (auto dest: prime-imp-coprime simp: *coprime-absorb-left*) also { have *: $(\prod p \in ?B. ?f p) = (\prod p \in ?B - ?C. ?f p) * (\prod p \in ?C. ?f p)$ using 1 n by (intro prod.subset-diff) (auto dest: dvd-imp-le) have $(\prod p \in ?B. ?f p) * ?P = (\prod p \in ?B - ?C. ?f p) * ((\prod p \in ?C. ?f p) *$ (P)**by** (*subst* *) (*simp add: mult-ac*) also have $(\prod p \in ?C. ?f p) * ?P = (\prod p \in ?C. 1)$ **by** (*subst prod.distrib* [*symmetric*], *rule prod.cong*) (insert s, auto simp: divide-simps powr-def exp-eq-1) also have $\ldots = 1$ by simpfinally have $(\prod p \in ?B - ?C. ?f p) = (\prod p \in ?B. ?f p) * ?P$ by simp

} also have $(\prod p \in ?B. ?f p) = (\prod p \le m. if prime p then ?f p else 1)$ by (intro prod.mono-neutral-cong-left) auto finally show ?case by (simp only: mult-ac) qed finally have ?g \longrightarrow Dirichlet-L n χ_0 s. moreover have ?g \longrightarrow ?P * zeta s by (intro tendsto-mult tendsto-const euler-product-zeta s) ultimately show Dirichlet-L n χ_0 s = ?P * zeta s by (rule LIMSEQ-unique) qed qed qed (insert $\langle s \neq 1 \rangle$ n, auto introl: holomorphic-intros holomorphic-Dirichlet-L-weak

 $open-halfspace-Re-gt\ exI[of - 2]\ connected-punctured-universe)$ qed (simp-all add: Dirichlet-L-def zeta-1) end

3.2 The non-vanishing for $\Re(s) \ge 1$

```
lemma coprime-prime-exists:
    assumes n > (0 :: nat)
    obtains p where prime p coprime p n
proof -
    from bigger-prime[of n] obtain p where p: prime p p > n by auto
    with assms have ¬p dvd n by (auto dest: dvd-imp-le)
    with p have coprime p n by (intro prime-imp-coprime)
    with that[of p] and p show ?thesis by auto
    ged
```

The case of the principal character is trivial, since it differs from the Riemann $\zeta(s)$ only in a multiplicative factor that is clearly non-zero for $\Re(s) \geq 1$.

theorem (in *residues-nat*) *Dirichlet-L-Re-ge-1-nonzero-principal*: assumes $Re \ s \ge 1 \ s \ne 1$ **shows** Dirichlet-L n (principal-dchar n) $s \neq 0$ proof have $(\prod p \mid prime \ p \land p \ dvd \ n. \ 1 - 1 \ / \ p \ powr \ s) \neq (0 :: complex)$ **proof** (subst prod-zero-iff) **from** *n* **show** finite {*p*. prime $p \land p$ dvd *n*} **by** (intro finite-prime-divisors) auto**show** $\neg(\exists p \in \{p. prime \ p \land p \ dvd \ n\}$. $1 - 1 \ / \ p \ powr \ s = 0)$ **proof** safe fix p assume p: prime p p dvd n and 1 - 1 / p powr s = 0hence norm (p powr s) = 1 by simp also have norm (p powr s) = real p powr Re s by (simp add: norm-powr-real-powr)finally show False using p assms by (simp add: powr-def prime-qt-0-nat) ged qed with zeta-Re-ge-1-nonzero[OF assms] show ?thesis by (simp add: Dirichlet-L-principal)

with zeta-Re-ge-1-nonzero[OF assms] **show** ?thesis **by** (simp add: Dirichlet-L-principal) **qed** The proof for non-principal character is quite involved and is typically very complicated and technical in most textbooks. For instance, Apostol [1] proves the result separately for real and non-real characters, where the non-real case is relatively short and nice, but the real case involves a number of complicated asymptotic estimates.

The following proof, on the other hand – like our proof of the analogous result for the Riemann ζ function – is based on Newman's book [4]. Newman gives a very short, concise, and high-level sketch that we aim to reproduce faithfully here.

context dcharacter begin theorem Dirichlet-L-Re-ge-1-nonzero-nonprincipal: assumes $\chi \neq \chi_0$ and $Re \ u \ge 1$ shows Dirichlet-L n $\chi \ u \ne 0$ proof (cases $Re \ u > 1$) include dcharacter-syntax case False define a where $a = -Im \ u$ from False and assms have $Re \ u = 1$ by simp hence [simp]: u = 1 - i * a by (simp add: a-def complex-eq-iff)

show ?thesis proof assume Dirichlet-L $n \ \chi \ u = 0$ hence zero: Dirichlet-L $n \ \chi \ (1 - i * a) = 0$ by simp define χ' where $[simp]: \chi' = inv$ -character χ

— We define the function Z(s), which is the product of all the Dirichlet L functions, and its Dirichlet series. Then, similarly to the proof of the non-vanishing of the Riemann ζ function for $\Re(s) \ge 1$, we define Q(s) = Z(s)Z(s+ia)Z(s-ia). Our objective is to show that the Dirichlet series of this function Q converges everywhere.

define Z where $Z = (\lambda s. \prod \chi \in dcharacters n. Dirichlet-L n \chi s)$ define Z-fds where Z-fds = $(\prod \chi \in dcharacters n. fds \chi)$ define Q where $Q = (\lambda s. Z s \ 2 * Z (s + i * a) * Z (s - i * a))$ define Q-fds where Q-fds = Z-fds $\ 2 * fds$ -shift (i * a) Z-fds * fds-shift (-i * a) Z-fds

let $?sings = \{1, 1 + i * a, 1 - i * a\}$

— Some preliminary auxiliary facts

define P where $P = (\lambda s. (\prod x \in \{p. prime \ p \land p \ dvd \ n\}. 1 - 1 / of-nat x powr s :: complex))$

have $\chi_0: \chi_0 \in dcharacters \ n \ by (auto simp: principal.dcharacter-axioms \ dcharacters-def)$

have [continuous-intros]: continuous-on A P for A unfolding P-def

by (intro continuous-intros) (auto simp: prime-gt-0-nat)

from this[of UNIV] have [continuous-intros]: isCont P s for s
by (auto simp: continuous-on-eq-continuous-at)

have $\chi: \chi \in dcharacters \ n \ \chi' \in dcharacters \ n \ using \ dcharacter-axioms$ by (auto simp add: dcharacters-def dcharacter.dcharacter-inv-character) from zero dcharacter.cnj-Dirichlet-L[of $n \ \chi \ 1 - i * a$] dcharacter-axioms have zero': Dirichlet-L $n \ \chi' \ (1 + i * a) = 0$ by simp

have $Z = (\lambda s. Dirichlet-L \ n \ \chi_0 \ s * (\prod \chi \in dcharacters \ n - \{\chi_0\}. Dirichlet-L \ n \ \chi \ s))$

unfolding Z-def using χ_0 by (intro ext prod.remove) auto

also have ... = $(\lambda s. P s * zeta s * (\prod \chi \in dcharacters n - {\chi_0}. Dirichlet-L n \chi s))$

by (*simp add: Dirichlet-L-principal P-def*)

finally have Z-eq: $Z = (\lambda s. P \ s \ast zeta \ s \ast (\prod \chi \in dcharacters \ n - \{\chi_0\}.$ Dirichlet-L n χ s)).

have Z-eq':
$$Z = (\lambda s. P s * zeta s * Dirichlet-L n \chi s *$$

 $(\prod \chi \in dcharacters n - \{\chi_0\} - \{\chi\}. Dirichlet-L n \chi s))$
if $\chi \in dcharacters n \chi \neq \chi_0$ for χ
proof (rule ext, goal-cases)
case (1 s)
from that have χ : $\chi \in dcharacters n$ by (simp add: dcharacters-def)
have $Z s = P s * zeta s *$
 $(\prod \chi \in dcharacters n - \{\chi_0\}. Dirichlet-L n \chi s)$ by (simp add: Z-eq)
also have $(\prod \chi \in dcharacters n - \{\chi_0\}. Dirichlet-L n \chi s) = Dirichlet-L n \chi$
*
 $(\prod \chi \in dcharacters n - \{\chi_0\} - \{\chi\}. Dirichlet-L n \chi s)$

($\prod \chi \in dcharacters \ n - \{\chi_0\} - \{\chi\}$. Dirichlet-L $n \ \chi \ s$) using assms χ that by (intro prod.remove) auto finally show ?case by (simp add: mult-ac) ged

— We again show that Q is locally bounded everywhere by showing that every singularity is cancelled by some zero. Since now, a can be zero, we do a case distinction here to make things a bit easier.

have Q-bigo-1: $Q \in O[at \ s](\lambda - . \ 1)$ for s

proof (cases a = 0)

case True

s

have (λs . Dirichlet-L $n \chi s - Dirichlet-L n \chi 1$) $\in O[at 1](\lambda s. s - 1)$ using χ assms n

by (*intro taylor-bigo-linear holomorphic-on-imp-differentiable-at*[*of - UNIV*] *holomorphic-intros*) (*auto simp: dcharacters-def*)

hence *: Dirichlet-L $n \ \chi \in O[at \ 1](\lambda s. \ s - 1)$ using zero True by simp have $Z = (\lambda s. \ P \ s \ * \ zeta \ s \ * \ Dirichlet-L \ n \ \chi \ s \ *$

 $(\prod \chi \in dcharacters \ n - \{\chi_0\} - \{\chi\}. \ Dirichlet-L \ n \ \chi \ s))$ using χ assms by (intro Z-eq') auto

also have $\ldots \in O[at \ 1](\lambda s. \ 1 * (1 \ / \ (s - 1)) * (s - 1) * 1)$ using $n \ \chi$ by (intro landau-o.big.mult continuous-imp-bigo-1 zeta-bigo-at-1 continuous-intros *)

(*auto simp*: *dcharacters-def*)

also have $(\lambda s::complex. 1 * (1 / (s - 1)) * (s - 1) * 1) \in \Theta[at 1](\lambda - 1)$ by (intro bigthetaI-cong) (auto simp: eventually-at-filter)

finally have Z-at-1: $Z \in O[at \ 1](\lambda - . \ 1)$. have $Z \in O[at \ s](\lambda$ -. 1) **proof** (cases s = 1) case False thus ?thesis unfolding Z-def using $n \chi$ by (intro continuous-imp-bigo-1 continuous-intros) (auto simp: dcharacters-def) qed (insert Z-at-1, auto) from $\langle a = 0 \rangle$ have $Q = (\lambda s. Z s * Z s * Z s * Z s)$ **by** (*simp add: Q-def power2-eq-square*) also have $\ldots \in O[at \ s](\lambda - . \ 1 \ * \ 1 \ * \ 1 \ * \ 1)$ **by** (*intro landau-o.big.mult*) *fact+* finally show ?thesis by simp next case False have bigo1: $(\lambda s. Z s * Z (s - i * a)) \in O[at 1](\lambda - . 1)$ **if** Dirichlet-L n χ (1 - i * a) = 0 a \neq 0 $\chi \in$ dcharacters n $\chi \neq \chi_0$ for a :: real and χ proof have $(\lambda s. Dirichlet-L \ n \ \chi \ (s - i * a) - Dirichlet-L \ n \ \chi \ (1 - i * a)) \in O[at$ $1](\lambda s. s - 1)$ using assms n that by (intro taylor-bigo-linear holomorphic-on-imp-differentiable-at[of - UNIV] holomorphic-intros) (auto simp: dcharacters-def) hence *: $(\lambda s. Dirichlet L n \chi (s - i * a)) \in O[at 1](\lambda s. s - 1)$ using that by simp have $(\lambda s. Z (s - i*a)) = (\lambda s. P (s - i*a) * zeta (s - i*a) * Dirichlet-L n$ $\chi (s - i * a)$ * $(\prod \chi \in dcharacters \ n - \{\chi_0\} - \{\chi\}. Dirichlet-L \ n \ \chi \ (s - i*a)))$ using that by (subst Z-eq'[of χ]) auto also have $\ldots \in O[at \ 1](\lambda s. \ 1 * 1 * (s - 1) * 1)$ unfolding *P*-def using that nby (intro landau-o.biq.mult continuous-imp-bigo-1 continuous-intros *) (auto simp: prime-gt-0-nat dcharacters-def) finally have $(\lambda s. Z (s - i * a)) \in O[at 1](\lambda s. s - 1)$ by simp moreover have $Z \in O[at \ 1](\lambda s. \ 1 * (1 \ / \ (s - 1)) * 1)$ unfolding Z-eq using n that by (intro landau-o.big.mult zeta-bigo-at-1 continuous-imp-bigo-1 continuous-intros) (*auto simp: dcharacters-def*) hence $Z \in O[at \ 1](\lambda s. \ 1 \ / \ (s - 1))$ by simp ultimately have $(\lambda s. Z s * Z (s - i * a)) \in O[at 1](\lambda s. 1 / (s - 1) * (s - i + a))$ -1))**by** (*intro landau-o*.*biq*.*mult*) also have $(\lambda s. 1 / (s - 1) * (s - 1)) \in \Theta[at 1](\lambda - . 1)$ **by** (*intro bigthetaI-cong*) (*auto simp add: eventually-at-filter*)

finally show ?thesis . qed have bigo1': $(\lambda s. Z s * Z (s + i * a)) \in O[at 1](\lambda - . 1)$ **if** Dirichlet-L n χ (1 - i * a) = 0 a \neq 0 $\chi \in$ dcharacters n $\chi \neq \chi_0$ for a :: real and χ proof – **from** that **interpret** dcharacter $n \in \chi$ by (simp-all add: dcharacters-def G-def) **from** bigo1[of inv-character χ -a] that cnj-Dirichlet-L[of 1 - i * a] **show** ?thesis **by** (*simp add: dcharacters-def dcharacter-inv-character*) qed have bigo2: $(\lambda s. Z s * Z (s - i * a)) \in O[at (1 + i * a)](\lambda - . 1)$ **if** Dirichlet-L n χ (1 - i * a) = 0 a \neq 0 $\chi \in$ dcharacters n $\chi \neq \chi_0$ for a :: real and χ proof have $(\lambda s. Z s * Z (s - i * a)) \in O[filtermap (\lambda s. s + i * a) (at 1)](\lambda - . 1)$ using $bigo1'[of \chi a]$ that by (simp add: mult.commute landau-o.big.in-filtermap-iff) also have filtermap ($\lambda s. s + i * a$) (at 1) = at (1 + i * a) using filtermap-at-shift[of -i * a 1] by simp finally show ?thesis . qed have bigo2': $(\lambda s. Z s * Z (s + i * a)) \in O[at (1 - i * a)](\lambda - . 1)$ **if** Dirichlet-L n χ (1 - i * a) = 0 a \neq 0 $\chi \in$ dcharacters n $\chi \neq \chi_0$ for a :: real and χ proof from that interpret dcharacter $n \ G \ \chi$ by (simp-all add: dcharacters-def G-def) **from** $bigo2[of inv-character \chi - a]$ that cnj-Dirichlet-L[of 1 - i * a] **show** ?thesis **by** (*simp add: dcharacters-def dcharacter-inv-character*) qed have Q-eq: $Q = (\lambda s. (Z s * Z (s + i * a)) * (Z s * Z (s - i * a)))$ **by** (*simp add: Q-def power2-eq-square mult-ac*) consider $s = 1 | s = 1 + i * a | s = 1 - i * a | s \notin$?sings by blast thus ?thesis proof cases case 1have $Q \in O[at \ 1](\lambda - . \ 1 \ * \ 1)$ unfolding Q-eq using assms zero zero' False χ by (intro landau-o.big.mult bigo1[of χ a] bigo1'[of χ a]; simp)+ with 1 show ?thesis by simp next case 2

have $Q \in O[at (1 + i * a)](\lambda - . 1 * 1)$ unfolding Q-eq using assms zero zero' False χ n by (intro landau-o.big.mult bigo2[of χ a] continuous-imp-bigo-1) (auto simp: Z-def dcharacters-def intro!: continuous-intros) with 2 show ?thesis by simp \mathbf{next} case 3 have $Q \in O[at (1 - i * a)](\lambda - . 1 * 1)$ unfolding Q-eq using assms zero zero' False χ n by (intro landau-o.big.mult bigo2'[of χ a] continuous-imp-bigo-1) (auto simp: Z-def dcharacters-def intro!: continuous-intros) with 3 show ?thesis by simp \mathbf{next} case 4thus ?thesis unfolding Q-def Z-def using n**by** (*intro continuous-imp-bigo-1 continuous-intros*) (auto simp: dcharacters-def complex-eq-iff) qed qed

— Again, we can remove the singularities from Q and extend it to an entire function.

have $\exists Q'. Q'$ holomorphic-on UNIV $\land (\forall z \in UNIV - ?sings. Q' z = Q z)$ using n by (intro removable-singularities Q-bigo-1)

 $(auto \ simp: \ Q-def \ Z-def \ dcharacters-def \ complex-eq-iff \ intro!: holomorphic-intros)$

then obtain Q' where Q': Q' holomorphic-on UNIV $\bigwedge z. z \notin ?sings \Longrightarrow Q'$ z = Q z by blast

— Q' constitutes an analytic continuation of the Dirichlet series of Q.
have eval-Q-fds: eval-fds Q-fds s = Q' s if Re s > 1 for s
proof have [simp]: dcharacter n χ if χ ∈ dcharacters n for χ
using that by (simp add: dcharacters-def)
from that have abs-conv-abscissa (fds χ) < ereal (Re s) if χ ∈ dcharacters
n for χ
using that by (intro le-less-trans[OF dcharacter.abs-conv-abscissa-weak[of n χ]]) auto
hence eval-fds Q-fds s = Q s using that
by (simp add: Q-fds-def Q-def eval-fds-mult eval-fds-power fds-abs-converges-mult

eval-fds-prod fds-abs-converges-prod dcharacter.Dirichlet-L-conv-eval-fds-weak
 fds-abs-converges-power eval-fds-zeta Z-fds-def Z-def fds-abs-converges)
 also from that have ... = Q' s by (subst Q') auto
 finally show ?thesis .
 qed

— Since the characters are completely multiplicative, the series for this logarithm can be rewritten like this:

define I where $I = (\lambda k. if [k = 1] \pmod{n}$ then totient n else 0 :: real) have *ln-Q-fds-eq*: fds-ln 0 Q-fds = fds (λk . of-real (2 * I k * mangoldt k / ln k * (1 + cos (a * ln k))))proof – have $nz: \chi$ (Suc 0) = 1 if $\chi \in dcharacters n$ for χ using dcharacter.Suc- $0[of n \chi]$ that by (simp add: dcharacters-def) note simps = fds-ln-mult [where l' = 0 and l'' = 0] fds-ln-power [where l'= 0fds-ln-prod[where $l' = \lambda$ -. θ] have fds-ln 0 Q-fds = $(\sum \chi \in dcharacters n. 2 * fds-ln 0 (fds \chi) +$ fds-shift (i * a) (fds-ln 0 (fds χ)) + fds-shift (-i * a) (fds-ln 0 (fds χ))) by (auto simp: Q-fds-def Z-fds-def simps nz sum.distrib sum-distrib-left) also have $\ldots = (\sum \chi \in dcharacters n.$ fds (λk . $\chi k * of-real$ (2 * mangoldt k / ln k * (1 + cos (a * ln k))))) $(\mathbf{is} \ (\sum \chi \in -. \ ?l \ \chi) = -)$ **proof** (*intro sum.cong refl, goal-cases*) case (1χ) then interpret dcharacter n G χ by (simp-all add: dcharacters-def G-def) have mult: completely-multiplicative-function (fds-nth (fds χ)) by (simp add: fds-nth-fds' mult.completely-multiplicative-function-axioms) have *: fds-ln 0 (fds χ) = fds (λn . χn * mangoldt n /_R ln (real n)) by (simp add: fds-ln-completely-multiplicative[OF mult] fds-nth-fds' fds-eq-iff) have ? $\chi = fds (\lambda k. \chi k * mangoldt k /_R \ln k * (2 + k powr (i * a) + k$ powr(-i * a)))by (unfold *, rule fds-eqI) (simp add: algebra-simps scaleR-conv-of-real numeral-fds) also have ... = fds (λk . $\chi k * 2 * mangoldt k /_R \ln k * (1 + cos (of-real(a)))$ * ln k))))**unfolding** cos-exp-eq **by** (intro fds-eqI) (simp add: powr-def algebra-simps) also have ... = fds (λk . $\chi k * of-real$ (2 * mangoldt k / ln k * (1 + cos (a * ln k))))unfolding cos-of-real by (simp add: field-simps scaleR-conv-of-real) finally show ?case . qed also have $\ldots = fds \ (\lambda k. \ (\sum \chi \in dcharacters \ n. \ \chi \ k) * of-real \ (2 * mangoldt$ k / ln k * $(1 + \cos(a * \ln k))))$ by (simp add: sum-distrib-right sum-divide-distrib scaleR-conv-of-real sum-distrib-left) also have $\ldots = fds \ (\lambda k. \ of-real \ (2 * I \ k * mangoldt \ k \ / \ ln \ k * (1 + cos \ (a + co$ * ln k))))by (intro fds-eqI, subst sum-dcharacters) (simp-all add: I-def algebra-simps) finally show ?thesis . qed The coefficients of that logarithm series are clearly nonnegative: have nonneq-dirichlet-series (fds-ln 0 Q-fds)

 \mathbf{proof}

show fds-nth (fds-ln 0 Q-fds) $k \in \mathbb{R}_{\geq 0}$ for k **proof** (cases k < 2) ${\bf case} \ {\it False}$ have cos: $1 + \cos x \ge 0$ for x :: realusing cos-qe-minus-one[of x] by linarith have fds-nth (fds-ln 0 Q-fds) k =of-real (2 * I k * mangoldt k / ln k * (1 + cos (a * ln k)))**by** (*auto simp: fds-nth-fds' ln-Q-fds-eq*) also have $\ldots \in \mathbb{R}_{>0}$ using *False* unfolding *I-def* **by** (subst nonneg-Reals-of-real-iff) (intro mult-nonneg-nonneg divide-nonneg-pos cos mangoldt-nonneg, auto) finally show ?thesis . **qed** (cases k; auto simp: ln-Q-fds-eq) qed Therefore *Q*-fds also has non-negative coefficients. hence nonneq: nonneq-dirichlet-series Q-fds **proof** (rule nonneq-dirichlet-series-lnD) have $(\prod x \in dcharacters n. x (Suc 0)) = 1$ by (intro prod.neutral) (auto simp: dcharacters-def dcharacter.Suc-0) thus $exp \ 0 = fds$ -nth Q-fds (Suc 0) by (simp add: Q-fds-def Z-fds-def) qed

— And by Pringsheim–Landau, we get that the Dirichlet series of Q converges everywhere.

have abs-conv-abscissa Q-fds ≤ 1 unfolding Q-fds-def Z-fds-def fds-shift-prod by (intro abs-conv-abscissa-power-leI abs-conv-abscissa-mult-leI abs-conv-abscissa-prod-le) (auto simp: dcharacters-def dcharacter.abs-conv-abscissa-weak)

with nonneg and eval-Q-fds and $\langle Q'$ holomorphic-on UNIV \rangle

have abscissa: abs-conv-abscissa Q-fds = $-\infty$

by (intro entire-continuation-imp-abs-conv-abscissa-MInfty[where g = Q'and c = 1])

(auto simp: one-ereal-def)

— Again, similarly to the proof for ζ , we select a subseries of Q. This time we cannot simply pick powers of 2, since 2 might not be coprime to n, in which case the subseries would simply be 1 everywhere, which is not helpful. However, it is clear that there *is* always some prime p that is coprime to n, so we just use the subseries Q that corresponds to powers of p.

from n obtain p where p: prime p coprime p n

using coprime-prime-exists[of n] by auto

define R-fds where R-fds = fds-primepow-subseries p Q-fds

have conv-abscissa R-fds \leq abs-conv-abscissa R-fds by (rule conv-le-abs-conv-abscissa) also have abs-conv-abscissa R-fds \leq abs-conv-abscissa Q-fds

unfolding *R*-fds-def by (rule abs-conv-abscissa-restrict)

also have $\ldots = -\infty$ by (simp add: abscissa)

finally have abscissa': conv-abscissa R- $fds = -\infty$ by simp

— The following function g(a, s) is the denominator in the Euler product expansion of the subseries of Z(s + ia). It is clear that it is entire and non-zero for

 $\Re(s) > 0$ and all real a.

define $g :: real \Rightarrow complex \Rightarrow complex$

where $g = (\lambda a \ s. (\prod \chi \in dcharacters \ n. (1 - \chi \ p * p \ powr \ (-s + i * of-real)))$ a)))) have *g*-nz: $g \ a \ s \neq 0$ if $Re \ s > 0$ for *s* a unfolding *g*-def

proof (*subst prod-zero-iff*[*OF finite-dcharacters*], *safe*)

fix χ assume $\chi \in dcharacters n$ and $*: 1 - \chi p * p powr (-s + i*a) = 0$ then interpret dcharacter n $G \chi$ by (simp-all add: dcharacters-def G-def) from p have real p > real 1 by (subst of-nat-less-iff) (auto simp: prime-gt-Suc-0-nat)

hence real p powr - Re s < real p powr 0

using p that by (intro powr-less-mono) auto

hence $\theta < norm (1 :: complex) - norm (\chi p * p powr (-s + i*a))$

using *p* **by** (*simp* add: *norm-mult norm norm-powr-real-powr*)

also have $\ldots \leq norm (1 - \chi p * p powr (-s + i*a))$

by (rule norm-triangle-ineq2)

finally show False by (subst (asm) *) simp-all

qed

have [holomorphic-intros]: g a holomorphic-on A for a A unfolding g-def using p by (intro holomorphic-intros)

— By taking Euler product expansions of every factor, we get

$$R(s) = \frac{1}{g(0,s)^2 g(a,s)g(-a,s)} = (1-2^{-s})^{-2}(1-2^{-s+ia})^{-1}(1-2^{-s-ia})^{-1}$$

for every s with $\Re(s) > 1$, and by analytic continuation also for $\Re(s) > 0$.

have eval-R: eval-fds R-fds $s = 1 / (g \ 0 \ s \ 2 * g \ a \ s * g \ (-a) \ s)$ (is - = ?f s) if Re s > 0 for s :: complexproof show ?thesis **proof** (rule analytic-continuation-open [where f = eval-fds R-fds]) **show** ?f holomorphic-on {s. Re s > 0} using p g - nz[of 0] g - nz[of a] g - nz[of a]-a**by** (*intro holomorphic-intros*) (*auto simp*: *g-nz*) \mathbf{next} fix z assume z: $z \in \{s. Re \ s > 1\}$ have [simp]: completely-multiplicative-function χ fds-nth (fds χ) = χ if $\chi \in dcharacters \ n$ for χ proof **from** that **interpret** dcharacter $n \ G \ \chi$ by (simp-all add: G-def dcharac*ters-def*) **show** completely-multiplicative-function χ fds-nth (fds χ) = χ **by** (simp-all add: fds-nth-fds' mult.completely-multiplicative-function-axioms) qed have [simp]: dcharacter $n \chi$ if $\chi \in$ dcharacters n for χ using that by (simp add: dcharacters-def) **from** that have abs-conv-abscissa (fds χ) < ereal (Re z) if $\chi \in$ dcharacters n for χ using that z by (intro le-less-trans[OF dcharacter.abs-conv-abscissa-weak[of $[n \chi]]) auto$

thus eval-fds R-fds z = ?f z using z p

 $\mathbf{by} \ (simp \ add: \ R-fds-def \ Q-fds-def \ Z-fds-def \ eval-fds-mult \ eval-fds-prod \ eval-fds-prod$

 $\mathit{fds-abs-converges-mult\ fds-abs-converges-power\ fds-abs-converges-prod\ g-def\ mult-ac}$

 $fds\-prime pow\-subservies\-euler\-product\-cm\ powr\-minus\ powr\-diff\ powr\-add\ prod\-dividef$

fds-abs-summable-zeta g-nz fds-abs-converges power-one-over divide-inverse [symmetric])

qed (insert that abscissa', auto intro!: exI[of - 2] convex-connected open-halfspace-Re-gt convex-halfspace-Re-gt holomorphic-intros)

qed

— We again have our contradiction: R(s) is entire, but the right-hand side has a pole at 0 since g(0,0) = 0.

show False

proof (*rule not-tendsto-and-filterlim-at-infinity*)

have g-limit: $(g \ a \longrightarrow g \ a \ 0)$ (at 0 within $\{s. \ Re \ s > 0\}$) for a proof –

have continuous-on UNIV $(g \ a)$ by (intro holomorphic-on-imp-continuous-on holomorphic-intros)

hence isCont (g a) 0 by (rule continuous-on-interior) auto

hence continuous (at 0 within {s. Re s > 0}) (g a) by (rule continuous-within-subset) auto

thus ?thesis by (auto simp: continuous-within) qed

have $((\lambda s. g \ 0 \ s \ 2 \ast g \ a \ s \ast g \ (-a) \ s) \longrightarrow g \ 0 \ 0 \ 2 \ast g \ a \ 0 \ast g \ (-a) \ 0)$

(at 0 within $\{s. Re \ s > 0\}$) by (intro tendsto-intros g-limit)

also have $g \ 0 \ 0 = 0$ unfolding g-def

proof (rule prod-zero)

from p and χ_0 show $\exists \chi \in dcharacters n. 1 - \chi p * of-nat p powr (-0 + i * of-real 0) = 0$

by (*intro* bexI[of - χ_0]) (*auto* simp: principal-dchar-def)

qed auto

moreover have eventually ($\lambda s. \ s \in \{s. \ Re \ s > 0\}$) (at 0 within $\{s. \ Re \ s > 0\}$)

by (*auto simp: eventually-at-filter*)

hence eventually ($\lambda s. g \ 0 \ s \ 2 \ * g \ a \ s \ * g \ (-a) \ s \neq 0$) (at 0 within {s. Re s > 0})

by eventually-elim (auto simp: g-nz)

ultimately have filterlim ($\lambda s. g \ 0 \ s \ 2 \ * g \ a \ s \ * g \ (-a) \ s$) (at 0)

(at 0 within $\{s. Re \ s > 0\}$) by (simp add: filterlim-at)

hence filterlim ?f at-infinity (at 0 within {s. $Re \ s > 0$ }) (is ?lim) by (intro filterlim-divide-at-infinity[OF tendsto-const]

tendsto-mult-filterlim-at-infinity) auto

also have ev: eventually (λs . Re s > 0) (at 0 within {s. Re s > 0})

by (auto simp: eventually-at intro!: exI[of - 1])

```
have ?lim \leftrightarrow filterlim (eval-fds R-fds) at-infinity (at 0 within {s. Re s > 0})
```

by (intro filterlim-cong refl eventually-mono[OF ev]) (auto simp: eval-R) finally show \mathbf{next} have continuous (at 0 within $\{s. Re \ s > 0\}$) (eval-fds R-fds) **by** (*intro continuous-intros*) (*auto simp: abscissa'*) **thus** ((eval-fds R-fds \longrightarrow eval-fds R-fds 0)) (at 0 within {s. Re s > 0}) by (auto simp: continuous-within) next have $\theta \in \{s. Re \ s \ge \theta\}$ by simp also have $\{s. Re \ s \ge 0\} = closure \{s. Re \ s > 0\}$ using closure-halfspace-gt[of 1::complex 0] by (simp add: inner-commute)finally have $\theta \in \ldots$. thus at 0 within $\{s. Re \ s > 0\} \neq bot$ **by** (subst at-within-eq-bot-iff) auto qed qed **qed** (fact Dirichlet-L-Re-gt-1-nonzero)

3.3 Asymptotic bounds on partial sums of Dirichlet L functions

The following are some bounds on partial sums of the L-function of a character that are useful for asymptotic reasoning, particularly for Dirichlet's Theorem.

```
lemma sum-upto-dcharacter-le:
 assumes \chi \neq \chi_0
 shows norm (sum-up to \chi x) \leq totient n
proof –
  have sum-up to \chi x = (\sum k \le nat \lfloor x \rfloor, \chi k) unfolding sum-up to-alt def
   by (intro sum.mono-neutral-left) auto
 also have norm \ldots \leq totient n
   by (rule sum-dcharacter-atMost-le) fact
 finally show ?thesis .
\mathbf{qed}
lemma Dirichlet-L-minus-partial-sum-bound:
  fixes s :: complex and x :: real
 assumes \chi \neq \chi_0 and Re \ s > 0 and x > 0
 defines \sigma \equiv Re \ s
           norm (sum-upto (\lambda n. \chi n * n powr -s) x - Dirichlet-L n \chi s) \leq
 shows
            real (totient n) * (2 + cmod s / \sigma) / x powr \sigma
proof (rule Lim-norm-ubound)
  from assms have summable (\lambda n. \chi n * of-nat n powr -s)
   by (intro summable-Dirichlet-L')
  with assms have (\lambda n. \chi n * of\text{-nat } n \text{ powr } -s) sums Dirichlet-L n \chi s
   using Dirichlet-L-conv-eval-fds[OF assms(1,2)]
   by (simp add: sums-iff eval-fds-def powr-minus divide-simps fds-nth-fds')
 hence (\lambda m. \sum k \leq m. \chi k * of\text{-nat } k \text{ powr } -s) \longrightarrow Dirichlet-L n \chi s
```

by (*simp add: sums-def' atLeast0AtMost*) **thus** (λm . sum-upto (λk . $\chi k * of$ -nat k powr -s) $x - (\sum k \le m, \chi k * of$ -nat k powr - s)) \rightarrow sum-upto ($\lambda k. \ \chi \ k * of-nat \ k \ powr \ -s$) $x - Dirichlet-L \ n \ \chi \ s$ **by** (*intro tendsto-intros*) \mathbf{next} define M where M = sum-upto χ have le: norm $(\sum n \in real - \{x < ... y\})$. $\chi n * of nat n powr - s)$ \leq real (totient n) * (2 + cmod s / σ) / x powr σ if xy: 0 < x x < y for x yproof from xy have I: $((\lambda t. M t * (-s * t powr (-s-1))))$ has-integral M y * of-real y powr - s - M x * of-real x powr - s - $(\sum n \in real - \{x < ... y\}, \chi n * of real (real n) powr - s)) \{x ... y\}$ unfolding M-def **by** (*intro partial-summation-strong* [of {}]) (auto introl: has-vector-derivative-real-field derivative-eq-intros continuous-intros) hence $(\sum n \in real - \{x < ... y\}, \chi n * real n powr - s) =$ M y * of-real y powr - s - M x * of-real x powr - s integral $\{x..y\}$ ($\lambda t. M t * (-s * t powr (-s-1))$) **by** (*simp add: has-integral-iff*) also have norm $\ldots \leq norm (M \ y * of - real \ y \ powr - s) + norm (M \ x * of - real)$ x powr - s) +norm (integral $\{x..y\}$ ($\lambda t. M t * (-s * t powr (-s-1))$)) by (intro order.trans[OF norm-triangle-ineq4] add-mono order.refl) also have norm $(M y * of - real y powr - s) \leq totient n * y powr - \sigma$ using xy assms unfolding norm-mult M-def σ -def by (intro mult-mono sum-upto-dcharacter-le) (auto simp: norm-powr-real-powr) also have $\ldots \leq totient \ n * x \ powr \ -\sigma$ using assms xy by (intro mult-left-mono powr-mono2') (auto simp: σ -def) also have norm $(M x * of - real x powr - s) \leq totient n * x powr - \sigma$ using xy assms unfolding norm-mult M-def σ -def by (intro mult-mono sum-upto-dcharacter-le) (auto simp: norm-powr-real-powr) also have norm (integral $\{x..y\}$ ($\lambda t. M t * (-s * of-real t powr (-s-1))$)) \leq integral $\{x..y\}$ ($\lambda t.$ real (totient n) * norm s * t powr ($-\sigma-1$)) **proof** (*rule integral-norm-bound-integral integrable-on-cmult-left*) show (λt . real (totient n) * norm s * t powr ($-\sigma - 1$)) integrable-on {x..y} using xy by (intro integrable-continuous-real continuous-intros) auto next fix t assume $t: t \in \{x..y\}$ have norm $(M \ t * (-s * of-real \ t \ powr \ (-s-1))) \leq$ real (totient n) * (norm s * t powr $(-\sigma - 1)$) unfolding norm-mult M-def σ -def using xy t assms by (intro mult-mono sum-upto-dcharacter-le) (auto simp: norm-mult norm-powr-real-powr) thus norm $(M \ t \ast (-s \ast of - real \ t \ powr \ (-s-1))) \le real \ (totient \ n) \ast norm \ s$ $* t powr (-\sigma - 1)$ **by** (*simp add: algebra-simps*) qed (insert I, auto simp: has-integral-iff)

also have ... = real (totient n) * norm s * integral {x..y} ($\lambda t. t powr(-\sigma-1)$) by simp also have $((\lambda t. t powr (-\sigma-1)) has-integral (y powr -\sigma / (-\sigma) - x powr -\sigma))$ $/(-\sigma)) \{x..y\}$ using xy assms **by** (*intro fundamental-theorem-of-calculus*) (auto intro!: derivative-eq-intros simp: has-real-derivative-iff-has-vector-derivative [symmetric] σ -def) hence integral $\{x...y\}$ $(\lambda t. t powr (-\sigma-1)) = y powr -\sigma / (-\sigma) - x powr -\sigma$ $/(-\sigma)$ by (simp add: has-integral-iff) also from assms have $\ldots \leq x \text{ powr } -\sigma / \sigma \text{ by } (simp \ add: \sigma \text{-}def)$ also have real (totient n) $*x powr -\sigma + real$ (totient n) $*x powr -\sigma + \sigma$ real (totient n) * norm s * (x powr $-\sigma / \sigma$) = real (totient n) $*(2 + norm s / \sigma) / x powr \sigma$ using xy by (simp add: field-simps powr-minus) finally show ?thesis by (simp add: mult-left-mono) qed **show** eventually (λm . norm (sum-upto (λk . $\chi k * of-nat k powr - s$) x - base k $(\sum k \leq m. \ \chi \ k * of-nat \ k \ powr \ - \ s))$ \leq real (totient n) * (2 + cmod s / σ) / x powr σ) at-top **using** eventually-gt-at-top[of nat |x|] **proof** eventually-elim case (elim m)have $(\sum k \leq m, \chi k * of-nat k powr - s) - sum-upto (\lambda k, \chi k * of-nat k powr)$ (-s) x = $(\sum k \in \{...m\} - \{k. \ 0 < k \land real \ k \le x\}, \chi \ k * of-nat \ k \ powr \ -s)$ unfolding sum-upto-def using $elim \langle x > 0 \rangle$ by (intro Groups-Big.sum-diff [symmetric]) (auto simp: nat-less-iff floor-less-iff) also have $\ldots = (\sum k \in \{\ldots m\} - \{k. real \ k \le x\}, \chi \ k * of nat \ k \ powr \ -s)$ by (intro sum.mono-neutral-right) auto also have $\{...m\} - \{k. real \ k \le x\} = real - `\{x < ...real \ m\}$ using $elim \langle x > 0 \rangle$ by (auto simp: nat-less-iff floor-less-iff not-less) also have norm $(\sum k \in \ldots \chi k * of nat k powr -s) \leq$ real (totient n) * (2 + cmod s / σ) / x powr σ using $elim \langle x > 0 \rangle$ by (intro le) (auto simp: nat-less-iff floor-less-iff) finally show ?case by (simp add: norm-minus-commute) qed $\mathbf{qed} \ auto$ **lemma** partial-Dirichlet-L-sum-bigo: fixes s :: complex and x :: realassumes $\chi \neq \chi_0 \ Re \ s > 0$ shows $(\lambda x. sum-upto (\lambda n. \chi n * n powr - s) x - Dirichlet-L n \chi s) \in O(\lambda x.$ x powr - sproof (rule bigoI) **show** eventually (λx . norm (sum-upto (λn . $\chi n * of-nat n powr -s$) x - Dirich $\begin{array}{l} let-L \ n \ \chi \ s) \\ \leq real \ (totient \ n) \ \ast \ (2 + norm \ s \ / \ Re \ s) \ \ast \ norm \ (of-real \ x \ powr \ - \ s)) \\ at-top \\ \textbf{using } eventually-gt-at-top[of \ 0] \\ \textbf{proof } eventually-elim \\ \textbf{case } (elim \ x) \\ \textbf{have } norm \ (sum-upto \ (\lambda n. \ \chi \ n \ \ast \ of-nat \ n \ powr \ -s) \ x \ - \ Dirichlet-L \ n \ \chi \ s) \\ \leq real \ (totient \ n) \ \ast \ (2 + norm \ s \ / \ Re \ s) \ / \ x \ powr \ Re \ s \\ \textbf{using } elim \ assms \ \textbf{by } \ (intro \ Dirichlet-L-minus-partial-sum-bound) \ auto \\ \textbf{thus } \ ?case \ \textbf{using } elim \ assms \\ \textbf{by } \ (simp \ add: \ norm-powr-real-powr \ powr-minus \ divide-simps \ norm-divide \\ del: \ div-mult-self1 \ div-mult-self2 \ div-mult-self3 \ div-mult-self4) \\ \textbf{qed} \\ \textbf{qed} \end{array}$

end

3.4 Evaluation of $L(\chi, 0)$

context residues-nat begin **lemma** Dirichlet-L-0-principal [simp]: Dirichlet-L n χ_0 $\theta = \theta$ proof have Dirichlet-L n χ_0 0 = $-1/2 * (\prod p \mid prime p \land p \, dvd n. 1 - 1 / p \, powr$ θ) **by** (*simp add: Dirichlet-L-principal prime-gt-0-nat*) **also have** $(\prod p \mid prime \ p \land p \ dvd \ n. \ 1 - 1 \ / \ p \ powr \ 0) = (\prod p \mid prime \ p \land p$ dvd n. 0 :: complex)**by** (*intro* prod.cong) (*auto* simp: prime-gt-0-nat) also have $(\prod p \mid prime \ p \land p \ dvd \ n. \ 0 :: complex) = 0$ using prime-divisor-exists [of n] n by (auto simp: card-gt-0-iff) finally show ?thesis by simp qed end context dcharacter begin lemma Dirichlet-L-0-nonprincipal: assumes nonprincipal: $\chi \neq \chi_0$ shows Dirichlet-L n $\chi 0 = -(\sum k=1..< n. of-nat \ k * \chi \ k) \ / of-nat \ n$ proof – have Dirichlet-L n χ 0 = ($\sum k=1..n$, χ k * (1 / 2 - of-nat k / of-nat n)) using assms n by (simp add: Dirichlet-L-conv-hurwitz-zeta-nonprincipal) also have $\ldots = -1/n * (\sum k=1..n. \text{ of-nat } k * \chi k)$ using assms by (simp add: algebra-simps sum-subtractf sum-dcharacter-block' sum-divide-distrib [symmetric]) also have $(\sum k=1..n. \text{ of-nat } k * \chi k) = (\sum k=1..< n. \text{ of-nat } k * \chi k)$ using n by (intro sum.mono-neutral-right) (auto simp: eq-zero-iff)

finally show eq: Dirichlet-L n $\chi 0 = -(\sum_{k=1..< n.} of-nat \ k * \chi k) / of-nat n$ by simp
qed

lemma Dirichlet-L-0-even [simp]: **assumes** χ (n - 1) = 1 **shows** Dirichlet-L $n \chi \ 0 = 0$ **proof** (cases $\chi = \chi_0$) **case** False **hence** Dirichlet-L $n \chi \ 0 = -(\sum k=Suc \ 0...< n. of-nat \ k * \chi \ k) \ / of-nat \ n$ **by** (simp add: Dirichlet-L-0-nonprincipal) **also** have ... = 0 **using** assms False **by** (subst even-dcharacter-linear-sum-eq-0) auto finally show Dirichlet-L $n \chi \ 0 = 0$. **qed** auto

lemma Dirichlet-L-0: Dirichlet-L $n \ \chi \ 0 = (if \ \chi \ (n - 1) = 1 \ then \ 0 \ else \ -(\sum k=1..< n. \ of-nat \ k * \chi k) \ / \ of-nat \ n)$ **by** (cases $\chi = \chi_0$) (auto simp: Dirichlet-L-0-nonprincipal) end

3.5 Properties of $L(\chi, s)$ for real χ

locale real-dcharacter = dcharacter + assumes real: $\chi \ k \in \mathbb{R}$ begin

lemma Im-eq-0 [simp]: Im $(\chi k) = 0$ using real[of k] by (auto elim!: Reals-cases)

lemma of-real-Re [simp]: of-real (Re (χk)) = χk by (simp add: complex-eq-iff)

lemma char-cases: $\chi \ k \in \{-1, \ 0, \ 1\}$ **proof** – **from** norm[of k] **have** Re ($\chi \ k$) $\in \{-1, 0, 1\}$ **by** (auto simp: cmod-def split: if-splits) **hence** of-real (Re ($\chi \ k$)) $\in \{-1, \ 0, \ 1\}$ **by** auto **also have** of-real (Re ($\chi \ k$)) = $\chi \ k$ **by** (simp add: complex-eq-iff) **finally show** ?thesis . **qed**

lemma cnj [simp]: cnj $(\chi k) = \chi k$ **by** (simp add: complex-eq-iff)

lemma inv-character-id [simp]: inv-character $\chi = \chi$ by (simp add: inv-character-def fun-eq-iff)

lemma Dirichlet-L-in-Reals:

assumes $s \in \mathbb{R}$ shows Dirichlet-L $n \ \chi \ s \in \mathbb{R}$ proof – have $cnj \ (Dirichlet-L \ n \ \chi \ s) = Dirichlet-L \ n \ \chi \ s$ using assms by (subst cnj-Dirichlet-L) (auto elim!: Reals-cases) thus ?thesis using Reals-cnj-iff by blast qed

The following property of real characters is used by Apostol to show the non-vanishing of $L(\chi, 1)$. We have already shown this in a much easier way, but this particular result is still of general interest.

lemma

assumes k: k > 0shows sum-char-divisors-ge: Re $(\sum d \mid d \ dvd \ k. \ \chi \ d) \ge 0$ (is Re $(?A \ k) \ge 0$) and sum-char-divisors-square-ge: is-square $k \Longrightarrow Re(\sum d \mid d \; dvd \; k, \; \chi \; d) \ge 1$ proof interpret sum: multiplicative-function ?A **by** (*fact mult.multiplicative-sum-divisors*) have A: ?A $k \in \mathbb{R}$ for k by (intro sum-in-Reals real) hence [simp]: Im $(?A \ k) = 0$ for k by (auto elim!: Reals-cases) have *: Re $(?A (p \cap m)) \ge (if even m then 1 else 0)$ if p: prime p for p m proof have sum-neg1: $(\sum i \le m. (-1) \ \hat{i}) = (if even m then 1 else (0::real))$ by $(induction \ m)$ auto from p have ?A $(p \uparrow m) = (\sum k \leq m. \chi (p \uparrow k))$ by (intro sum.reindex-bij-betw [symmetric] bij-betw-prime-power-divisors) also have $Re \ldots = (\sum k \le m. Re(\chi p) \land k)$ by (simp add: mult.power) also have $\ldots > (if even m then 1 else 0)$ **using** sum-neg1 char-cases[of p] **by** (auto simp: power-0-left) finally show ?thesis . qed have *: Re $(?A (p \cap m)) \ge 0$ even $m \Longrightarrow Re (?A (p \cap m)) \ge 1$ if prime p for p musing *[of p m] that by (auto split: *if-splits*) **have** eq: Re $(?A \ k) = (\prod p \in prime factors k. Re (?A (p \cap multiplicity p k))))$ using $k \ A$ by (subst sum.prod-prime-factors) (auto simp: Re-prod-Reals) **show** Re $(\sum d \mid d \; dvd \; k. \; \chi \; d) \geq 0$ by (subst eq, intro prod-nonneg ball *) auto assume is-square kthen obtain m where m: $k = m^2$ by (auto elim!: is-nth-powerE) have even (multiplicity p k) if prime p for p using k that unfolding mby (subst prime-elem-multiplicity-power-distrib) (auto introl: Nat.gr0I) thus $Re (\sum d \mid d \ dvd \ k. \ \chi \ d) \ge 1$

 $\label{eq:substeq} \begin{array}{l} \mathbf{by} \ (subst \ eq, \ intro \ prod-ge-1 \ ball I \ *) \ auto \\ \mathbf{qed} \end{array}$

end

4 Dirichlet's Theorem on primes in arithmetic progressions

theory Dirichlet-Theorem imports Dirichlet-L-Functions Bertrands-Postulate.Bertrand Landau-Symbols.Landau-More

begin

We can now turn to the proof of the main result: Dirichlet's theorem about the infinitude of primes in arithmetic progressions.

There are previous proofs of this by John Harrison in HOL Light [3] and by Mario Carneiro in Metamath [2]. Both of them strive to prove Dirichlet's theorem with a minimum amount of auxiliary results and definitions, whereas our goal was to get a short and simple proof of Dirichlet's theorem built upon a large library of Analytic Number Theory.

At this point, we already have the key part – the non-vanishing of $L(1, \chi)$ – and the proof was relatively simple and straightforward due to the large amount of Complex Analysis and Analytic Number Theory we have available. The remainder will be a bit more concrete, but still reasonably concise.

First, we need to re-frame some of the results from the AFP entry about Bertrand's postulate a little bit.

4.1 Auxiliary results

The AFP entry for Bertrand's postulate already provides a slightly stronger version of this for integer values of x, but we can easily extend this to real numbers to obtain a slightly nicer presentation.

```
lemma sum-upto-mangoldt-le:

assumes x \ge 0

shows sum-upto mangoldt x \le 3 / 2 * x

proof –

have sum-upto mangoldt x = psi (nat \lfloor x \rfloor)

by (simp add: sum-upto-altdef psi-def atLeastSucAtMost-greaterThanAtMost)

also have ... \le 551 / 256 * ln 2 * real (nat \lfloor x \rfloor)

by (rule psi-ubound-log)

also have ... \le 3 / 2 * real (nat \lfloor x \rfloor)

using Bertrand.ln-2-le by (intro mult-right-mono) auto

also have ... \le 3 / 2 * x using assms by linarith

finally show ?thesis .

qed
```

end

We can also, similarly, use the results from the Bertrand's postulate entry to show that the sum of $\ln p/p$ over all primes grows logarithmically.

lemma *Mertens-bigo*:

 $(\lambda x. (\sum p \mid prime \ p \land real \ p \le x. \ ln \ p \ / \ p) - ln \ x) \in O(\lambda$ -. 1) **proof** (*intro bigoI*[*of - 8*] *eventually-mono*[*OF eventually-gt-at-top*[*of 1*]], *goal-cases*) case (1 x)have $|(\sum p \mid prime \ p \land p \le nat \ \lfloor x \rfloor. \ ln \ p \ / \ p) - ln \ x| =$ $|(\sum p \mid prime \ p \land p \le nat \ \lfloor x \rfloor. \ ln \ p \ / \ p) - ln \ (nat \ \lfloor x \rfloor) - (ln \ x - ln \ (nat \ \lfloor x \rfloor)) - (ln \ x - ln \ x - ln \ (ln \ x - ln \ (ln \ x - ln \ x - ln \ (ln \ x - ln \ x - ln \ (ln \ x - ln \ x - ln \ x - ln \ (ln \ x - ln \ x - ln \ (ln \ x - ln \ (ln \ x - ln \ (ln \ x - ln \$ |x|))|by simp also have $\ldots \leq |(\sum p \mid prime \ p \land p \leq nat \ \lfloor x \rfloor. \ ln \ p \ / \ p) - ln \ (nat \ \lfloor x \rfloor)| + |ln|$ x - ln (nat |x|)by (rule abs-triangle-ineq4) also have $|(\sum p \mid prime \ p \land p \le nat \ \lfloor x \rfloor. \ ln \ p \ / \ p) - ln \ (nat \ |x|)| \le 7$ using 1 by (intro Mertens) auto also have $|\ln x - \ln (nat \lfloor x \rfloor)| = \ln x - \ln (nat \lfloor x \rfloor)$ using 1 by (intro abs-of-nonneg) auto also from 1 have $\ldots \leq (x - nat |x|) / nat |x|$ by (intro ln-diff-le) auto also have $\ldots \leq (x - nat \lfloor x \rfloor) / 1$ using 1 by (intro divide-left-mono) auto also have $\ldots = x - nat |x|$ by simp also have $\ldots \leq 1$ using 1 by linarith **also have** $(\sum p \mid prime \ p \land p \leq nat \ \lfloor x \rfloor. \ ln \ p \ / \ p) = (\sum p \mid prime \ p \land real \ p \leq nat \ \lfloor x \rfloor. \ ln \ p \ / \ p) = (\sum p \mid prime \ p \land real \ p \leq nat \ \lfloor x \rfloor. \ ln \ p \ / \ p) = (\sum p \mid prime \ p \land real \ p \leq nat \ \lfloor x \rfloor. \ p \)$ x. ln p / p) using 1 by (intro sum.cong refl) (auto simp: le-nat-iff le-floor-iff) finally show ?case by simp qed

4.2 The contribution of the non-principal characters

The estimates in the next two sections are partially inspired by John Harrison's proof of Dirichlet's Theorem [3].

We first estimate the growth of the partial sums of

$$-L'(1,\chi)/L(1,\chi) = \sum_{k=1}^{\infty} \chi(k) \frac{\Lambda(k)}{k}$$

for a non-principal character χ and show that they are, in fact, bounded, which is ultimately a consequence of the non-vanishing of $L(1,\chi)$ for nonprincipal χ .

```
context dcharacter begin
```

 $\operatorname{context}$

includes no vec-lambda-syntax and dcharacter-syntax fixes L assumes nonprincipal: $\chi \neq \chi_0$ defines $L \equiv Dirichlet-L \ n \ \chi \ 1$

begin

lemma Dirichlet-L-nonprincipal-mangoldt-bound-aux-strong: assumes x: x > 0shows norm (L * sum-upto (λk . χk * mangoldt k / k) x - sum-upto (λk . χk * ln k / k x $\leq 9 / 2 * real (totient n)$ proof define B where B = 3 * real (totient n) have sum-up to $(\lambda k, \chi k * \ln k / k) x = sum-up to (\lambda k, \chi k * (\sum d \mid d \, dv d \, k))$ mangoldt d) / k) x**by** (*intro sum-upto-cong*) (*simp-all add: mangoldt-sum*) also have ... = sum-upto (λk . $\sum d \mid d \; dvd \; k$. $\chi \; k * mangoldt \; d \; / \; k$) x **by** (*simp add: sum-distrib-left sum-distrib-right sum-divide-distrib*) also have ... = sum-upto (λk . sum-upto (λd . χ (d * k) * mangoldt k / (d * k)) (x / real k)) xby (rule sum-upto-sum-divisors) also have ... = sum-upto (λk . $\chi k * mangoldt k / k * sum-upto (<math>\lambda d$. $\chi d / d$) (x / real k)) x**by** (*simp add: sum-upto-def sum-distrib-left mult-ac*) also have $L * sum-upto (\lambda k. \chi k * mangoldt k / k) x - ... =$ sum-upto (λk . (L - sum-upto (λd . χ d / d) (x / real k)) * (χ k * mangoldt k / k) x unfolding ring-distribs by (simp add: sum-upto-def sum-subtractf sum-distrib-left mult-ac) also have *norm*... \leq *sum-upto* (λk . B / x * mangoldt k) x unfolding *sum-upto-def* **proof** (*rule sum-norm-le, goal-cases*) case (1 k)have norm $((L - sum - upto (\lambda d. \chi d / of - nat d) (x / k)) * \chi k * (mangoldt k)$ $(of-nat k)) \leq$ $\stackrel{-}{(B}$ * real k / x) * 1 * (mangoldt k / real k) unfolding norm-mult norm-divide **proof** (*intro mult-mono divide-left-mono*) show norm $(L - sum - upto (\lambda d. \chi d / d) (x / k)) \leq B * real k / x$ using Dirichlet-L-minus-partial-sum-bound [OF nonprincipal, of $1 \times k$] $1 \times k$ by (simp add: powr-minus B-def L-def divide-simps norm-minus-commute) qed (insert 1, auto introl: divide-nonneg-pos mangoldt-nonneg norm-le-1 simp: B-def) also have $\ldots = B / x * mangoldt k$ using 1 by simp finally show ?case by (simp add: sum-upto-def mult-ac) qed also have $\ldots = B / x * sum$ -upto mangoldt x unfolding sum-upto-def sum-distrib-left by simp also have $\ldots \leq B / x * (3 / 2 * x)$ using x by (intro mult-left-mono sum-upto-mangoldt-le) (auto simp: B-def) also have $\dots = 9 / 2 * real$ (totient n) using x by (simp add: B-def) finally show ?thesis by (simp add: B-def) qed

lemma *Dirichlet-L-nonprincipal-mangoldt-aux-bound*:

 $(\lambda x. \ L * sum-upto \ (\lambda k. \ \chi \ k * mangoldt \ k \ / \ k) \ x - sum-upto \ (\lambda k. \ \chi \ k * ln \ k \ / \ k)$ $x) \in O(\lambda$ -. 1) by (intro bigoI[of - 9 / 2 * real (totient n)] eventually-mono[OF eventually-gt-at-top[of 0]])(use Dirichlet-L-nonprincipal-mangoldt-bound-aux-strong in simp-all) **lemma** *nonprincipal-mangoldt-bound*: $(\lambda x. sum-upto \ (\lambda k. \ \chi \ k * mangoldt \ k \ / \ k) \ x) \in O(\lambda - 1)$ (is ?lhs \in -) proof have [simp]: $L \neq 0$ using nonprincipal unfolding L-def by (intro Dirichlet-L-Re-ge-1-nonzero-nonprincipal) auto have fds-converges (fds-deriv (fds χ)) 1 using conv-abscissa-le-0[OF nonprincipal] **by** (*intro fds-converges-deriv*) (*auto intro: le-less-trans*) hence summable $(\lambda n. -(\chi n * \ln n / n))$ by (auto simp: fds-converges-def fds-deriv-def scaleR-conv-of-real fds-nth-fds' algebra-simps) hence summable $(\lambda n, \chi n * \ln n / n)$ by (simp only: summable-minus-iff) from summable-imp-convergent-sum-up to [OF this] obtain c where $(sum-upto \ (\lambda n. \ \chi \ n * ln \ n \ / \ n) \longrightarrow c) \ at-top \ by \ blast$ hence *: sum-upto (λk . $\chi k * ln k / k$) $\in O(\lambda$ -. 1) unfolding sum-upto-def

by (*intro bigoI-tendsto*[of - -c]) *auto*

from sum-in-bigo[OF Dirichlet-L-nonprincipal-mangoldt-aux-bound *]

have $(\lambda x. L * sum-upto (\lambda k. \chi k * mangoldt k / k) x) \in O(\lambda-. 1)$ by (simp add: L-def)

thus ?thesis by simp qed

end end

4.3 The contribution of the principal character

Next, we turn to the analogous partial sum for the principal character and show that it grows logarithmically and therefore is the dominant contribution.

```
context residues-nat
begin
context
includes no vec-lambda-syntax and dcharacter-syntax
begin
```

lemma principal-dchar-sum-bound:

 $(\lambda x. (\sum p \mid prime \ p \land real \ p \leq x. \ \chi_0 \ p * (ln \ p \ / \ p)) - ln \ x) \in O(\lambda -. 1)$ proof -

have fin [simp]: finite {p. prime $p \land real p \le x \land Q p$ } for Q xby (rule finite subset[of [nat | x|]]) (sets simp: le pat iff le floor

by (rule finite-subset[of - {...nat $\lfloor x \rfloor$ }]) (auto simp: le-nat-iff le-floor-iff)

from $fin[of - \lambda - . True]$ have [simp]: $finite \{p. prime p \land real p \leq x\}$ for x by $(simp \ del: fin)$

define P :: complex where $P = (\sum p \mid prime \ p \land p \ dvd \ n. of real \ (ln \ p \ / \ p))$

have $(\lambda x. (\sum p \mid prime \ p \land real \ p \leq x. \ \chi_0 \ p \ast (ln \ p \ / \ p)) - ln \ x) \in$

 $\Theta(\lambda x. \text{ of-real } ((\sum p \mid prime \ p \land real \ p \le x. \ln p \ / \ p) - \ln x) - P) \ (\mathbf{is} \ - \in \Theta(?f))$

proof (*intro* bigthetaI-cong eventually-mono[OF eventually-gt-at-top[of real n]], goal-cases)

case (1 x)

have *: { $p. prime \ p \land real \ p \le x$ } =

 $\{p. \ prime \ p \land real \ p \le x \land p \ dvd \ n\} \cup \{p. \ prime \ p \land real \ p \le x \land \neg p \ dvd \ n\}$

 $(\mathbf{is} - = ?A \cup ?B)$ by *auto*

have $(\sum p \mid prime p \land real p \le x. of real (ln p / p)) =$

 $(\sum p \in ?A. of-real (ln p / p)) + (\sum p \in ?B. complex-of-real (ln p / p))$

 $\mathbf{by} \; (subst \; *, \; subst \; sum.union-disjoint) \; auto$

also from 1 have $?A = \{p. prime \ p \land p \ dvd \ n\}$ using n by (auto dest: dvd-imp-le)

also have $(\sum p \in \dots \text{ of-real } (\ln p / p)) = P$ by (simp add: P-def)also have $(\sum p \in ?B. \text{ of-real } (\ln p / p)) =$

 $(\sum p \mid prime \ p \land real \ p \le x. \ \chi_0 \ p \ast (ln \ p \ / \ p))$ by (intro sum.mono-neutral-cong-left)

(auto simp: principal-dchar-def prime-gt-0-nat coprime-absorb-left prime-imp-coprime)

finally show ?case using 1 by (simp add: Ln-of-real)

 \mathbf{qed}

also have $?f \in O(\lambda$ -. of-real 1)

by (rule sum-in-bigo, subst landau-o.big.of-real-iff, rule Mertens-bigo) auto

finally show ?thesis by simp

qed

lemma *principal-dchar-sum-bound'*:

 $(\lambda x. sum-upto \ (\lambda k. \ \chi_0 \ k * mangoldt \ k \ / \ k) \ x - Ln \ x) \in O(\lambda -. 1)$ proof -

have $(\lambda x. sum-upto \ (\lambda k. \ \chi_0 \ k * mangoldt \ k \ / \ k) \ x -$

 $(\sum p \mid prime \ p \land real \ p \leq x. \ \chi_0 \ p \ast (ln \ p / p))) \in O(\lambda$ -. 1)

proof (intro bigoI[of - 3] eventually-mono[OF eventually-gt-at-top[of 0]], goal-cases)
case (1 x)

have norm (complex-of-real ($\sum k \mid real \ k \leq x \land coprime \ k \ n.$ mangoldt $k \ / \ k$)

of-real
$$(\sum p \mid prime \ p \land p \in \{k. \ real \ k \leq x \land coprime \ k \ n\}$$
. In $p \mid p$

 $p)) \leq 3$

unfolding of-real-diff [symmetric] norm-of-real

by (rule Mertens-mangoldt-versus-ln[where $n = nat \lfloor x \rfloor])$

(insert n, auto simp: Suc-le-eq le-nat-iff le-floor-iff introl: Nat.gr0I)

also have complex-of-real $(\sum k \mid real \ k \leq x \land coprime \ k \ n. mangoldt \ k \ / \ k) = sum-upto (\lambda k. \chi_0 \ k * mangoldt \ k \ / \ k) \ x$

unfolding sum-upto-def of-real-sum using n

by (intro sum.mono-neutral-cong-left) (auto simp: principal-dchar-def introl:

\mathbf{qed}

4.4 The main result

We can now show the main result by extracting the primes we want using the orthogonality relation on characters, separating the principal part of the sum from the non-principal ones and then applying the above estimates.

lemma Dirichlet-strong: assumes coprime h n**shows** $(\lambda x. (\sum p \mid prime \ p \land [p = h] \pmod{n} \land real \ p \leq x. \ln p \ / p) - \ln x \ /$ totient n) $\in O(\lambda$ -. 1) (is $(\lambda x. sum - (?A x) - -) \in -)$ proof from assms obtain h' where h': $[h * h' = Suc \ 0] \pmod{n}$ **by** (subst (asm) coprime-iff-invertible-nat) blast let $?A' = \lambda x$. $\{p. p > 0 \land real p \leq x \land [p = h] \pmod{n}\}$ let $?B = dcharacters n - \{\chi_0\}$ have [simp]: $\chi_0 \in dcharacters n$ **by** (*auto simp: dcharacters-def principal.dcharacter-axioms*) have $(\lambda x. \text{ of-nat } (\text{totient } n) * (\sum p \in ?A' x. \text{ mangoldt } p / p) - \ln x) \in$ $\Theta(\lambda x. sum-upto (\lambda k. \chi_0 \ k * mangoldt \ k \ / \ k) \ x - \ln x +$ $(\sum \chi \in \mathcal{B}. \ \chi \ h' * sum-upto \ (\lambda k. \ \chi \ k * mangoldt \ k \ / \ k) \ x))$ $(\mathbf{is} - \in \Theta(\mathcal{P}f))$ **proof** (*intro bigthetaI-cong eventually-mono*[OF eventually-gt-at-top[of 0]], goal-cases) case (1 x)have of-nat (totient n) $* (\sum p \in A' x. of-real (mangoldt p / p) :: complex) =$ $(\sum p \in A' x. of-nat (totient n) * of-real (mangoldt p / p))$ **by** (subst sum-distrib-left) simp-all also have ... = sum-upto (λk . $\sum \chi \in dcharacters n$. χ (h' * k) * (mangoldt k / k)) xunfolding *sum-upto-def* **proof** (*intro sum.mono-neutral-cong-left ballI*, *goal-cases*) case (3 p)have $[h' * p \neq 1] \pmod{n}$ proof assume $[h' * p = 1] \pmod{n}$

hence $[h * (h' * p) = h * 1] \pmod{n}$ by (rule cong-scalar-left) hence $[h = h * h' * p] \pmod{n}$ by (simp add: mult-ac cong-sym) also have $[h * h' * p = 1 * p] \pmod{n}$ using h' by (intro cong-scalar-right) auto finally have $[p = h] \pmod{n}$ by $(simp \ add: \ cong-sym)$ with 3 show False by auto qed thus ?case by (auto simp: sum-dcharacters sum-divide-distrib [symmetric] sum-distrib-right [symmetric]) \mathbf{next} case (4 p)hence $[p * h' = h * h'] \pmod{n}$ by (intro cong-scalar-right) auto also have $[h * h' = 1] \pmod{n}$ using h' by simp finally have $[h' * p = 1] \pmod{n}$ by $(simp \ add: \ mult-ac)$ thus ?case using h' 4by (auto simp: sum-dcharacters sum-divide-distrib [symmetric] sum-distrib-right [symmetric]) qed auto also have $\ldots = (\sum \chi \in dcharacters \ n. \ sum-upto \ (\lambda k. \ \chi \ (h' * k) * (mangoldt \ k$ (k)(x)unfolding sum-upto-def by (rule sum.swap) also have ... = $(\sum \chi \in dcharacters \ n. \ \chi \ h' * sum-upto \ (\lambda k. \ \chi \ k * (mangoldt \ k$ (k)(x)unfolding *sum-upto-def* by (intro sum.cong refl) (auto simp: dcharacters-def dcharacter.mult sum-distrib-left mult-ac) also have ... = $\chi_0 h' * sum$ -upto ($\lambda k. \chi_0 k * (mangoldt k / k)$) x + $(\sum \chi \in ?B. \ \chi \ h' * sum-upto \ (\lambda k. \ \chi \ k * (mangoldt \ k \ / \ k)) \ x)$ by (subst sum.remove [symmetric]) (auto simp: sum-distrib-left) also have coprime h' nusing h' by (subst coprime-iff-invertible-nat, subst (asm) mult.commute) auto hence $\chi_0 h' = 1$ **by** (*simp add: principal-dchar-def*) finally show ?case using n 1 by (simp add: Ln-of-real) qed also have $?f \in O(\lambda$ -. complex-of-real 1) **proof** (rule sum-in-bigo[OF - big-sum-in-bigo], goal-cases) case 1 from principal-dchar-sum-bound' show ?case by simp \mathbf{next} case (2χ) then interpret dcharacter n G χ by (simp-all add: G-def dcharacters-def) from 2 have $\chi \neq \chi_0$ by *auto* thus ?case unfolding of-real-1 by (intro landau-o.big.mult-in-1 nonprincipal-mangoldt-bound) auto ged finally have $*: (\lambda x. real (totient n) * (\sum p \in ?A' x. mangoldt p / p) - ln x) \in$ $O(\lambda - . 1)$

by (*subst* (*asm*) *landau-o.big.of-real-iff*)

have $(\lambda x. real (totient n) * ((\sum p \in A x. ln p / p) - (\sum p \in A' x. mangoldt p / p)))$ $p))) \in O(\lambda - 1)$ **proof** (*intro landau-o.big.mult-in-1*) show $(\lambda x. (\sum p \in ?A x. \ln p / p) - (\sum p \in ?A' x. mangoldt p / p)) \in O(\lambda -. 1)$ unfolding landau-o.big.of-real-iff **proof** (*intro bigoI*[*of - 3*] *eventually-mono*[*OF eventually-gt-at-top*[*of 0*]], *goal-cases*) case (1 x)have $|(\sum p \in A' x. mangoldt p / p) - (\sum p | prime p \land p \in A' x. ln p / p)|$ ≤ 3 by (rule Mertens-mangoldt-versus-ln[where n = nat |x|])(auto simp: le-nat-iff le-floor-iff) also have $\{p, prime \ p \land p \in ?A' x\} = ?A x$ by (auto simp: prime-gt-0-nat) finally show ?case by (simp add: abs-minus-commute) qed qed auto from sum-in-bigo(1)[OF * this]have $(\lambda x. \text{ totient } n * (\sum p \in A x. \ln p / p) - \ln x) \in O(\lambda - 1)$ **by** (*simp add: field-simps*) also have $(\lambda x. \text{ totient } n * (\sum p \in ?A x. \ln p / p) - \ln x) = (\lambda x. \text{ totient } n * ((\sum p \in ?A x. \ln p / p) - \ln x / \text{ totient } n))$ using *n* by (intro ext) (auto simp: field-simps) also have $\ldots \in O(\lambda - 1) \iff$?thesis using n **by** (*intro landau-o.big.cmult-in-iff*) *auto* finally show ?thesis . qed

It is now obvious that the set of primes we are interested in is, in fact, infinite.

theorem Dirichlet: assumes coprime h n shows infinite {p. prime $p \land [p = h] \pmod{n}$ } proof assume finite {p. prime $p \land [p = h] \pmod{n}$ } then obtain K where K: {p. prime $p \land [p = h] \pmod{n}$ } $\subseteq \{... < K\}$ by (auto simp: finite-nat-iff-bounded) have eventually ($\lambda x. (\sum p \mid prime p \land [p = h] \pmod{n} \land real p \le x. \ln p / p) = (\sum p \mid prime p \land [p = h] \pmod{n}. \ln p / p)$) at-top using eventually-ge-at-top[of real K] by eventually-elim (intro sum.cong, use K in auto) hence ($\lambda x. (\sum p \mid prime p \land [p = h] \pmod{n} \land real p \le x. \ln p / p)$) $\in \Theta(\lambda -. (\sum p \mid prime p \land [p = h] \pmod{n}. \ln p / p))$ by (intro bigthetaI-cong) auto

also have $(\lambda - (\sum p \mid prime \ p \land [p = h] \pmod{n}. \ln p / p)) \in O(\lambda - 1)$ by simp finally have $(\lambda x. (\sum p \mid prime \ p \land [p = h] \pmod{n} \land real \ p \le x. \ln p / p)) \in O(\lambda - 1)$.

from sum-in-bigo(2)[OF this Dirichlet-strong[OF assms]] and n show False by simp

 \mathbf{qed}

In the future, one could extend this result to more precise estimates of the distribution of primes in arithmetic progressions in a similar way to the Prime Number Theorem.

end end end

References

- T. M. Apostol. Introduction to Analytic Number Theory. Undergraduate Texts in Mathematics. Springer-Verlag, 1976.
- [2] M. Carneiro. Formalization of the prime number theorem and dirichlet's theorem. In Proceedings of the 9th Conference on Intelligent Computer Mathematics (CICM 2016), pages 10–13, 2016.
- [3] J. Harrison. A formalized proof of Dirichlet's theorem on primes in arithmetic progression. *Journal of Formalized Reasoning*, 2(1):63–83, 2009.
- [4] D. Newman. Analytic Number Theory. Number 177 in Graduate Texts in Mathematics. Springer, 1998.