# Dirichlet $L$-functions and Dirichlet's Theorem 

Manuel Eberl

September 13, 2023


#### 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 $\mathfrak{R}(s) \geq 1$. 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 $p \equiv h(\bmod n)$.


## Contents

1 Multiplicative 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 $\mathfrak{R}(s) \geq 1$ ..... 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 $\chi$ ..... 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<br>imports<br>Complex-Main<br>Finitely-Generated-Abelian-Groups.Finitely-Generated-Abelian-Groups<br>begin<br>notation integer-mod-group ( $Z$ )

### 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 \(\chi=(\lambda a . c n j(\chi a))\)
lemma inv-character-principal [simp]: inv-character (principal-char \(G\) ) \(=\) princi-
pal-char \(G\)
    by (simp add: inv-character-def principal-char-def fun-eq-iff)
lemma inv-character-inv-character \([\operatorname{simp}]:\) inv-character \((\) inv-character \(\chi)=\chi\)
    by (simp add: inv-character-def)
lemma eval-inv-character: inv-character \(\chi j=c n j(\chi j)\)
    by (simp add: inv-character-def)
bundle character-syntax
begin
notation principal-char \(\left(\chi_{01}\right)\)
end
locale character \(=\) finite-comm-group +
    fixes \(\chi\) :: ' \(a \Rightarrow\) complex
```

assumes char-one-nz: $\chi \mathbf{1} \neq 0$
assumes char-eq-0: $\quad a \notin$ carrier $G \Longrightarrow \chi a=0$
assumes char-mult $[$ simp $]: a \in$ carrier $G \Longrightarrow b \in \operatorname{carrier} G \Longrightarrow \chi(a \otimes b)=\chi$
$a * \chi b$
begin

### 1.2 Basic properties

lemma char-one [simp]: $\chi \mathbf{1}=1$
proof-
from char-mult $\left[\begin{array}{ll}\text { of } & \mathbf{1}\end{array}\right]$ 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 $[\operatorname{simp}]: a \in \operatorname{carrier} G \Longrightarrow \chi(a[ \urcorner k)=\chi a^{\wedge} k$
by (induction $k$ ) auto
lemma char-root:
assumes $a \in$ carrier $G$
shows $\quad \chi a{ }^{\wedge}$ ord $a=1$
proof -
from assms have $\chi a{ }^{\wedge}$ ord $a=\chi(a[\uparrow$ ord $a)$
by (subst char-power) auto
also from fin and assms have $a[\uparrow$ ord $a=\mathbf{1}$ by (intro pow-ord-eq-1) auto
finally show ?thesis by simp
qed
lemma char-root':
assumes $a \in$ carrier $G$
shows $\quad \chi a^{\wedge}$ order $G=1$
proof -
from assms have $\chi a^{\wedge}$ order $G=\chi(a[\uparrow$ order $G)$ by simp
also from fin and assms have $a[\uparrow$ order $G=\mathbf{1}$ 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 ( $\chi a){ }^{\wedge}$ order $G=$ norm ( $\chi$ a ${ }^{\wedge}$ order $G$ ) by (simp add: norm-power)
also from True have $\chi a^{\wedge}$ order $G=1$ by (rule char-root')
finally have norm ( $\chi a)^{\wedge}$ order $G=1$ ^ order $G$ by simp
hence norm $(\chi a)=1$ by (subst (asm) power-eq-iff-eq-base) auto
with True show ?thesis by auto
next
case False
thus ?thesis by (auto simp: char-eq-0)
qed

```
lemma char-eq-0-iff: \chi a=0\longleftrightarrowa\not< carrier G
proof -
    have \chi a=0 \longleftrightarrow norm ( }\chi\mathrm{ a) = 0 by simp
    also have ...\longleftrightarrowa\not\incarrier 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 Gk
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 Gk
        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 \otimesa=1 by simp
    also have \chi .. = 1 by simp
    also from assms have \chi(inv a\otimesa)=\chi(inv a)*\chia
        by (intro char-mult) auto
    finally have *: \chi (inv a)* \chi a=1.
    thus \chi(inv a)= inverse ( }\chi\mathrm{ a) by (auto simp: divide-simps)
    also from mult-inv-character[of a] and assms have inverse ( }\chi\mathrm{ a) =cnj ( }\chi\mathrm{ | a)
    by (auto simp add: inv-character-def principal-char-def divide-simps mult.commute)
    finally show \chi (inv a) = cnj (\chia).
qed
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 \(G \chi^{\prime} \bigwedge x . x \in \operatorname{carrier} G \Longrightarrow \chi x=\chi^{\prime} x\) shows \(\quad \chi=\chi^{\prime}\)
proof
fix \(x::\) ' \(a\)
```

```
    show \chix= \chi' }
    using assms by (cases x carrier G) (auto simp: character.char-eq-0)
qed
lemma character-mult [intro]:
    assumes character G \chi character G \chi
    shows character G (\lambdax. \chix* 㿟 x)
proof -
    interpret \chi: character G \chi by fact
    interpret }\mp@subsup{\chi}{}{\prime}\mathrm{ : character G }\mp@subsup{\chi}{}{\prime}\mathrm{ by fact
    show ?thesis by standard (auto simp: \chi.char-eq-0)
qed
```

lemma character-inv-character-iff [simp]: character $G$ (inv-character $\chi) \longleftrightarrow$ char-
acter $G \chi$
proof
assume character $G$ (inv-character $\chi$ )
from character.inv-character [OF this] show character $G \chi$ by simp
qed (auto simp: character.inv-character)
definition characters :: (' $a$, ' $b$ ) monoid-scheme $\Rightarrow$ ( ${ }^{\prime} a \Rightarrow$ complex) set where
characters $G=\{\chi$. 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 \(G=\left(\right.\) carrier \(=\) characters \(G\), monoid.mult \(=\left(\begin{array}{ll}\lambda & \chi_{1} \chi_{2}\end{array}\right.\). \(\chi_{1}\)
\(\left.k * \chi_{2} k\right)\),
        one \(=\) principal-char \(G D\)
lemma carrier-Characters: carrier (Characters \(G\) ) \(=\) characters \(G\)
    by (simp add: Characters-def)
lemma one-Characters: one (Characters \(G\) ) \(=\) principal-char \(G\)
    by (simp add: Characters-def)
lemma mult-Characters: monoid.mult (Characters \(G) \chi_{1} \chi_{2}=\left(\begin{array}{ll}\lambda a . & \chi_{1}\end{array} a * \chi_{2} a\right)\)
    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\()\) '
                            \(P i_{E}(\) carrier \(G)\left(\lambda-.\left\{z . z^{\wedge}\right.\right.\) order \(\left.\left.G=1\right\}\right)(\) is \(-\subseteq\) ? \(h\) '?Chars \()\)
    proof (intro subsetI, goal-cases)
        case (1 \(\chi\) )
        then interpret \(\chi\) : character \(G \chi\) by (simp add: characters-def)
        have ? \(h(\) restrict \(\chi(\) carrier \(G)) \in ? ~\) ' ? Chars
            by (intro imageI) (auto simp: \(\chi\).char-root')
    also have ? \(h(\) restrict \(\chi(\) carrier \(G))=\chi\) by (simp add: fun-eq-iff \(\chi\). char-eq- 0 )
    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 \(\chi \chi^{\prime}\) assume \(*: \chi \in\) carrier (Characters \(\left.G\right) \chi^{\prime} \in\) carrier (Characters \(G\) )
    from \(*\) interpret \(\chi\) : character \(G \chi\) by (simp-all add: characters-def carrier-Characters)
    from * interpret \(\chi^{\prime}\) : character \(G \chi^{\prime}\) by (simp-all add: characters-def car-
rier-Characters)
    have character \(G\left(\lambda k . \chi k * \chi^{\prime} k\right)\)
        by standard (insert \(*\), simp-all add: \(\chi\).char-eq-0 one-Characters
                            mult-Characters characters-def carrier-Characters)
    thus \(\chi \otimes_{\text {Characters } G} \chi^{\prime} \in\) carrier (Characters \(G\) )
    by (simp add: characters-def one-Characters mult-Characters carrier-Characters)
next
    have character \(G\) (principal-char \(G\) ) ..
    thus \(\mathbf{1}_{\text {Characters }} G \in\) carrier (Characters \(G\) )
    by (simp add: characters-def one-Characters mult-Characters carrier-Characters)
next
    fix \(\chi\) assume \(*: ~ \chi \in\) carrier (Characters \(G\) )
    from \(*\) interpret \(\chi\) : character \(G \chi\) by (simp-all add: characters-def carrier-Characters)
    show \(\mathbf{1}_{\text {Characters }} G \otimes^{\otimes}\) Characters \(G \chi=\chi\) and \(\chi \otimes_{\text {Characters }} G \mathbf{1}_{\text {Characters }} G\)
\(=\chi\)
    by (simp-all add: principal-char-def fun-eq-iff \(\chi\).char-eq-0 one-Characters mult-Characters)
next
    have \(\chi \in\) Units (Characters \(G\) ) if \(\chi \in\) carrier (Characters \(G\) ) for \(\chi\)
    proof -
        from that interpret \(\chi\) : character \(G \chi\) by (simp add: characters-def car-
rier-Characters)
    have \(\chi \otimes_{\text {Characters } G}\) inv-character \(\chi=\mathbf{1}_{\text {Characters }} G\) and
                inv-character \(\chi \otimes_{\text {Characters } G} \chi=\mathbf{1}_{\text {Characters }} G\)
    by (simp-all add: \(\chi\).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 $G=1$
shows $\quad \chi=$ principal-char $G$
proof -
from assms have card (carrier $G-\{\mathbf{1}\})=0$
by (subst card-Diff-subset) (auto simp: order-def)
hence carrier $G-\{\mathbf{1}\}=\{ \}$
by (subst (asm) card-0-eq) auto
hence carrier $G=\{\mathbf{1}\}$ by auto
thus ?thesis
by (intro ext) (simp-all add: principal-char-def char-eq-0)
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)
lemma (in character) inv-Characters: inv Characters $G \chi=$ inv-character $\chi$
proof -
interpret Characters: finite-comm-group Characters $G$..
have character $G \chi$..
thus ?thesis
by (intro Characters.inv-equality)
(auto simp: characters-def mult-inv-character mult-ac
carrier-Characters one-Characters mult-Characters)
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 $G \Longrightarrow$ inv $_{\text {Characters }} G \chi=$ 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 $G$
and gen-fixes-char: $\llbracket$ character $G a ;$ character $G b ;$ a gen $=b$ gen $\Longrightarrow a=b$
and unity-root-induce-char: $z^{\wedge}$ order $G=1 \Longrightarrow$ character $G$ (induce-char $z$ ) proof -
interpret $C$ : finite-comm-group Characters $G$ using finite-comm-group-Characters
define $n$ where $n=$ order $G$
hence $n: n>0$ using order-gt-0 by presburger
from $n$-def have nog: $n=$ ord gen using ord-gen-is-group-order by simp
have $x n z: x \neq 0$ if $x^{\wedge} n=1$ for $x::$ complex using $n(1)$ that by (metis zero-neq-one zero-power)
have $m$ : $x$ powi $m=x$ powi $(m \bmod n)$ if $x \wedge n=1$ for $x:$ :complex and $m::$ int using powi-mod[OF that n].
show $c f$ : character $G$ (induce-char $x$ ) if $x: x^{\wedge} n=1$ for $x$
proof
show induce-char $x \mathbf{1} \neq 0$ using $x n z[O F$ that $]$ unfolding ic by auto
show induce-char $x a=0$ 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 $x b$
if $a \in$ 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 b
proof -
have $x$ powi get-exp gen $(a \otimes b)=x$ powi $(($ get-exp gen $a+$ get-exp gen $b)$ $\bmod n)$
using $m[O F x]$ get-exp-mult-mod $[O F$ that $] n$-def ord-gen-is-group-order by metis
also have $\ldots=x$ powi (get-exp gen $a+$ get-exp gen $b$ ) using $m[O F x]$ by presburger
finally show ?thesis by (simp add: power-int-add xnz[OF x])
qed
thus ?thesis using that unfolding ic by simp
qed
qed
define get- $c$ where $g c:$ get- $c=\left(\lambda c:::^{\prime} a \Rightarrow\right.$ complex. c gen $)$
have biji: bij-betw induce-char $\left\{z . z^{\wedge} n=1\right\}$ (characters $G$ )
and bijg: bij-betw get-c (characters $G$ ) $\left\{z . z^{\wedge} n=1\right\}$
proof (intro bij-betwI[of -- get-c])
show iin: induce-char $\in\left\{z . z^{\wedge} n=1\right\} \rightarrow$ characters $G$ using cf unfolding
characters-def
by blast
show gi: get-c (induce-char $x$ ) $=x$ if $x \in\left\{z . z^{\wedge} n=1\right\}$ for $x$
proof (cases $n=1$ )
case True
with that have $x=1$ by force
thus ?thesis unfolding ic gc by simp
next
case False
have $x: x^{\wedge} n=1$ using that by blast
have $x$ powi get-exp gen gen $=x$
proof -
have $x$ powi get-exp gen gen $=x$ powi $($ get-exp gen gen mod $n)$ using $m[O F$
$x]$ by blast
moreover have $($ get-exp gen gen $\bmod n)=1$
proof -
have $1=1 \bmod$ int $n$ using False $n$ by auto
also have $\ldots=$ get-exp gen 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 show $x$ powi get-exp gen gen $=x$ by simp
qed
thus ?thesis unfolding ic gc by simp
qed
show gin: get-c $\in$ characters $G \rightarrow\left\{z . z^{\wedge} n=1\right\}$
proof -
have False if get-c $c^{\wedge} n \neq 1$ 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 $y c: y \in$ 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=ya 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 .. = y gen powi ((1::int) mod n)
        using get-exp-self[OF gen-closed] nog by argo
    also have ... = y gen powi 1 using False n by simp
    finally have yg: y gen powi get-exp gen gen = y gen by simp
    thus ?case.
    case (step x)
    have y gen powi get-exp gen (x\otimesgen)=y gen powi (get-exp gen ( }x\otimes\mathrm{ gen )
mod n)
        using m[OF ygo] by blast
        also have }\ldots=y\mathrm{ y gen powi ((get-exp gen }x+\mathrm{ get-exp gen gen) mod n)
        using get-exp-mult-mod[OF step(1) gen-closed, unfolded nog[symmetric]]
by argo
            also have ... = y gen powi (get-exp gen x + get-exp gen gen) using m[OF
ygo] by presburger
            also have ... = 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\otimesgen) using step (1) by simp
            finally 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^ 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 bigg[unfolded gc characters-def bij-betw-def inj-on-def] show }a=b\mathrm{ 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 \(\chi\) where finite-cyclic-group (Characters \(G\) ) \(\chi\)
proof -
    interpret \(C\) : finite-comm-group Characters \(G\) by (rule finite-comm-group-Characters)
    define \(n\) where \(n: n=\) order \(G\)
    hence \(n n z: n \neq 0\) by blast
    from \(n\) have nog: \(n=\) ord gen using ord-gen-is-group-order by simp
    obtain \(x::\) complex where \(x: x^{\wedge} n=1 \bigwedge m . \llbracket 0<m ; m<n \rrbracket \Longrightarrow x^{\wedge} m \neq 1\)
        using true-nth-unity-root by blast
```

have $x n z: x \neq 0$ using $x n$ by (metis order-gt-0 zero-neq-one zero-power)
have $m$ : x powi $m=x$ powi $(m \bmod n)$ for $m::$ int
using powi-mod $[$ OF $x(1)] n n z$ by blast
let ?f $=(\lambda a$. if $a \in$ carrier $G$ then $x$ powi (get-exp gen a) else 0)
have $c f$ : character $G$ ?f using unity-root-induce-char[OF $x(1)$ [unfolded n]].
have fpow: (?f [ $]_{\text {Characters } G} m$ ) $a=x$ powi $(($ get-exp gen $a) * m)$
if $a \in$ carrier $G$ for $a::^{\prime} a$ and $m:: n a t$
using 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 -
fix $m a$ :: nat
have $x$ powi $((1+$ int ma) * get-exp gen a)
$=x$ powi $($ get-exp gen $a+$ int $m a *$ get-exp gen $a) \wedge 0 \neq x$
by (simp add: comm-semiring-class.distrib xnz)
then show $x$ powi (get-exp gen $a *$ int ma) * x powi get-exp gen a

$$
=x \text { powi }(\text { get-exp gen } a *(1+\text { int ma }))
$$

by (simp add: mult.commute power-int-add)
qed
thus ?case using $s$ by simp
qed $\operatorname{simp}$
interpret cyclic-group Characters $G$ ?f
proof (intro C.element-ord-generates-cyclic)
show $f c: ? f \in$ carrier (Characters $G$ ) using cf carrier-Characters $[o f ~ G]$ char-acters-def by fast
from $x n n z$ have fno: ?f [] Characters $G m \neq \mathbf{1}_{\text {Characters } G}$ if $0<m m<n$ for $m$
proof (cases $n=1$ )
case False
have $1_{\text {Characters } G}$ gen $=1$ unfolding Characters-def principal-char-def
using that by simp
moreover have (?f [ $]_{\text {Characters } G} m$ ) gen $\neq 1$
proof -
have (?f [ $\left.]_{\text {Characters } G} m\right)$ gen $=x$ powi $(($ get-exp gen gen $) * m)$ using
fpow by blast
also have $\ldots=(x$ powi $($ get-exp gen gen $)) ~ \wedge m$ by (simp add: power-int-mult)
also have $\ldots=x^{\wedge} m$
proof -
have $x$ powi $($ get-exp gen gen $)=x$ powi $(($ get-exp gen gen $) \bmod n)$ using
$m$ by blast
moreover have $(($ get-exp gen gen $) \bmod n)=1$
proof -
have $1=1$ mod int $n$ using False nnz by simp
also have $\ldots$. get-exp gen 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 }\leqn\mathrm{ 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.
lemma (in finite-cyclic-group) Characters-iso:
$G \cong$ Characters $G$
proof -
from finite-cyclic-group-Characters obtain $f$ where $f$ : finite-cyclic-group (Characters $G) f$.
then interpret $C$ : finite-cyclic-group Characters $G f$.
have cyclic-group (Characters $G$ ) $f$ by unfold-locales
from iso-cyclic-groups-same-order[OF this order-Characters[symmetric]] show
?thesis.
qed
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 $H$ 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 $C G$ : 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 0$) \neq 0$ 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 $x\left(a \otimes_{H} b\right)=$ ? $x a *$ ? $x b$ if $a \in$ carrier $H b \in$ carrier $H$ for $a b$ using that by auto
qed
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 $x$ using $f$-in $[$ OF that $]$.
show ?f $\left(x \otimes_{\text {Characters } G} y\right)=$ ?f $x \otimes_{\text {Characters } H}$ ?f $y$
if $x \in$ carrier (Characters $G) y \in$ carrier (Characters $G$ ) for $x y$
proof -
interpret $x$ : character $G x$ using that characters-def carrier-Characters by blast
interpret $y$ : character $G y$ using that characters-def carrier-Characters by blast show ?thesis using that mult-Characters $[$ of $G]$ mult-Characters $[o f ~ H]$ by auto
qed
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 simp
then 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 $g$-in: ? $g x \in$ carrier (Characters $G$ ) if $x \in$ carrier (Characters $H$ ) for $x$
proof (unfold carrier-Characters characters-def, rule, unfold-locales)
interpret character $H x$ using that characters-def carrier-Characters by
blast
show (if $\mathbf{1}_{G} \in$ carrier $G$ then $(x \circ f) \mathbf{1}_{G}$ else 0$) \neq 0$ using $f$ iso-iff by
auto
show $\bigwedge a . a \notin$ carrier $G \Longrightarrow($ if $a \in$ carrier $G$ then $(x \circ f)$ a else 0$)=0$ by $\operatorname{simp}$
show ? $g x\left(a \otimes_{G} b\right)=? g x a * ? g x b$ if $a \in \operatorname{carrier} G b \in$ 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 }
    proof -
    interpret character Hx using x characters-def carrier-Characters by blast
    have ?f (?g x) a=x a if a: a\not\in carrier H for a using a char-eq-O[OF a]
by auto
    moreover have ?f (?g x) a=x a if a:a\incarrier H for a
    proof -
            from a have inv-into (carrier H)g(ga)=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(?f x)=x if x: x\in carrier (Characters G) for }
    proof -
        interpret character G x using x characters-def carrier-Characters by blast
        have ?g(?f x) a = x a if a: a\not\in carrier G for a using a char-eq-0 [OF a]
by auto
            moreover have ?g (?f x) a=x a if a: a\incarrier G for a using a f-def
            proof -
            from a have g(inv-into (carrier H)ga)=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
        qed
    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\inI
    and I: finite I
    defines g:g\equiv(\lambdac. (\lambdai\inI. (\lambdaa.c c((\lambdai\inI. 1 Gs i) (i:=a)))))
    shows character (Gs i) (gxi)
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: \i.i\inI\Longrightarrowgroup (Gs i) using DirProds-group-imp-groups[OF DP.is-group]
    show ?thesis
    proof(unfold-locales)
    have }(\lambdai\inI.\mp@subsup{\mathbf{1}}{Gs i}{})=(\lambdai\inI.\mp@subsup{\mathbf{1}}{Gs i}{})(i:=\mp@subsup{\mathbf{1}}{Gs i}{})\mathrm{ using i by force
    thus g x i 1 Gs i\not=0 using i g DirProds-one'"[of Gs I] xc.char-one-nz by auto
    show g x i a=0 if a: a\not\incarrier (Gs i) for a
    proof -
        from a i have ((\lambdai\inI. 1 Gs i)}(i:=a))\not\incarrier (DirProds Gs I)
            unfolding DirProds-def by force
        from xc.char-eq-0[OF this] show ?thesis using ig by auto
    qed
    show g xi (a* Gs ib)=gxia*gxib
        if ab:a\incarrier (Gs i) b\in\operatorname{carrier (Gs i) for ab}
    proof -
        have g x i (a \otimes Gs i b)
                =x ((\lambdai\inI.\mp@subsup{\mathbf{1}}{Gs i}{*})(i:=a)\mp@subsup{\otimes}{\mathrm{ DirProds Gs I }}{}(\lambdai\inI.\mp@subsup{\mathbf{1}}{Gs i}{*})(i:=b))
        proof -
            have ((\lambdai\inI. 1 Gs i)(i:=a) \otimes DirProds Gs I (\lambdai\inI. 1 Gs i) (i:= b))
                =((\lambdai\inI. 1 1Gs i)(i:=(a\otimes Gs i}\mp@subsup{}{}{b}))
            proof -
                have ((\lambdai\inI. 1 1 Gs i)}(i:=a)\mp@subsup{\otimes}{\mathrm{ DirProds Gs I }}{}(\lambdai\inI.\mp@subsup{\mathbf{1}}{Gs i}{*})(i:=b))
                    = ((\lambdai\inI. 1 1Gs i)}(i:=(a\mp@subsup{\otimes}{Gsi}{}\mp@subsup{|}{}{\prime})))
                for }
                proof (cases j \inI)
                    case True
                        from allg[OF True] interpret Gj: group Gs j.
                show ?thesis using ab True i unfolding DirProds-mult by simp
                next
                    case False
                                    then show ?thesis unfolding DirProds-mult using i by fastforce
                qed
                thus ?thesis by fast
            qed
            thus ?thesis using ig by auto
        qed
```



```
        proof -
            have ac: ((\lambdai\inI. 1 Gs i})(i:=a))\in\operatorname{carrier (DirProds Gs I)
        unfolding DirProds-def using ab i monoid.one-closed[OF group.is-monoid[OF
allg]] by force
            have bc: ((\lambdai\inI. 1 1Gs i)(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 .
        qed
        also have ... = gxia*gxib using ig 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 $g: g \equiv\left(\lambda I .\left(\lambda c .\left(\lambda i \in I .\left(\lambda a . c\left(\left(\lambda i \in I .1_{G s}\right)(i:=a)\right)\right)\right)\right)\right)$
shows ( $\lambda$ e. if $e \in$ carrier(DirProds Gs I) then $\prod i \in I$. ( $\left.g I x i\right)(e$ i) else 0) $=x$
(is ? $g x=x$ )
proof
show ? $g x e=x e$ for $e$
proof (cases e carrier (DirProds Gs I))
case True
show ?thesis using $I x$ assms (1) True unfolding $g$
proof (induction I arbitrary: x e rule: finite-induct)
case empty
interpret $D P$ : 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
next
case $j$ : (insert $j$ I)
interpret $D P$ : 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 allg: $\bigwedge i . i \in($ insert $j I) \Longrightarrow$ group (Gs i)
using DirProds-group-imp-groups[OF DP.is-group].
have $e 1 c: e\left(j:=\mathbf{1}_{G s} j\right) \in \operatorname{carrier}($ DirProds Gs (insert $\left.j I)\right)$
using $j(6)$ monoid.one-closed[OF group.is-monoid[OF allg[of j]]]
unfolding DirProds-def PiE-def Pi-def by simp
have $e 2 c:\left(\lambda i \in(\right.$ insert $\left.j I) . \mathbf{1}_{G s} i\right)(j:=e j) \in$ 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 $e=e\left(j:=\mathbf{1}_{G s} j\right) \otimes_{\text {DirProds }}$ Gs $_{(\text {insert } j I)}\left(\lambda i \in(\right.$ insert $\left.j I) . \mathbf{1}_{G s}\right)(j:=$
e j)
proof -
have $e k$

$$
=\left(e\left(j:=\mathbf{1}_{G s}\right) \otimes_{\text {DirProds } G s ~}^{\text {insert j I })} \text { ( } \lambda i \in(\text { insert j I }) . \mathbf{1}_{G s}\right)(j:=e
$$

j)) $k$
for $k$
proof(cases $k \in($ insert $j$ I))
case $k$ : True
from allg $[O F k]$ interpret Gk: group Gs $k$.
from allg $[$ of $j]$ interpret $G j$ : group Gs $j$ by simp
from $k$ show ?thesis unfolding comp-mult $[O F k]$ 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 $x e=x\left(e\left(j:=\mathbf{1}_{G s} j\right)\right) * x\left(\left(\lambda i \in(\right.\right.$ insert $\left.\left.j I) . \mathbf{1}_{G s} i\right)(j:=e j)\right)$
using xc.char-mult $[$ OF e1c e2c] by argo
also have $\ldots=\left(\prod i \in I . g(\right.$ insert $\left.j I) x i(e i)\right) * g($ insert $j I) x j(e j)$
proof -
have $x\left(e\left(j:=\mathbf{1}_{G s} j\right)\right)=\left(\prod i \in I . g(\right.$ insert $\left.j I) x i(e i)\right)$
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 $p \in \operatorname{carrier(DirProds~Gs~} I)$ then $x\left(p\left(j:=\mathbf{1}_{\text {Gs }}\right)\right)$ else 0
have cx2: character (DirProds Gs I) ? $x$
proof
show ? $\mathbf{1}_{1 \text { DirProds Gs } I} \neq 0$
proof -
have $\mathbf{1}_{\text {DirProds }}$ Gs $I\left(j:=\mathbf{1}_{G s} j\right)=\mathbf{1}_{\text {DirProds }}$ 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 ? $x\left(a \otimes_{\text {DirProds Gs } I} b\right)=? x a * ? x b$
if ab: $a \in$ carrier (DirProds Gs I) $b \in$ carrier (DirProds Gs I) for $a b$
proof -
have $a c: a\left(j:=\mathbf{1}_{G s} j\right) \in \operatorname{carrier}$ (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 $b c: b\left(j:=\mathbf{1}_{G s} j\right) \in \operatorname{carrier}($ 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 $m:\left(\left(a \otimes_{\text {DirProds }} G s{ }^{b}\right)\left(j:=\mathbf{1}_{G s} j\right)\right)$

$$
=\left(a\left(j:=\mathbf{1}_{G s} j\right) \otimes_{\text {DirProds Gs (insert j } I)} b\left(j:=\mathbf{1}_{\text {Gs }}\right)\right)
$$

proof -

```
have}((a\mp@subsup{\otimes}{\mathrm{ DirProds Gs I }}{
```



```
    if h:h\in(insert j I) for h
proof(cases h=j)
    case True
    interpret Gj: group Gs j using allg[of j] by blast
        from True comp-mult[OF h, of Gs a(j:== 1 Gs j)b(j:= \mathbf{1}}\mp@subsup{\mp@code{Gs j})]}{}{\prime
show ?thesis
            by auto
        next
            case False
            interpret Gj: group Gs h using allg[OF h] .
            from False h comp-mult[OF h, of Gs a(j:= \mathbf{1}}\mp@subsup{G}{\mathrm{ s j ) b (j := 1}}{Gs j)]
                comp-mult[of h I Gs a b]
            show ?thesis by auto
            qed
            moreover have ((a* DirProds Gs I }b)(j:=\mp@subsup{\mathbf{1}}{Gs j}{)})
```



```
                    if h:h\not\in(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}}\mp@subsup{\mp@code{Gs j })}{}{)}
                =x (a(j:=\mp@subsup{\mathbf{1}}{Gs j))*x (b(j:= 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 \i\inI.gI ?x i ((e(j:=undefined)) i)
        else 0) = ?x (e(j:=undefined ))
    using eu j(2) unfolding g by fast
with eu have (\prodi\inI.g I (\lambdap. if p\incarrier (DirProds Gs I)
                                    then x (p(j:=\mp@subsup{\mathbf{1}}{\mathrm{ Gs j})}{})
                                    else 0) i ((e(j:= undefined)) i)) =x (e(j:=
1 (Gs j))
by \(\operatorname{simp}\)
moreover have \(g I\) ( \(\lambda a\). if \(a \in\) carrier (DirProds Gs \(I\) )
                                    then x (a(j:=\mp@subsup{\mathbf{1}}{Gs j}{}))
                    else 0) i ((e(j:= undefined)) i)=g(insert jI) x i (e i)
    if i:i\inI for }
proof -
    have (\lambdai\inI. 1 1Gs 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 $\operatorname{simp}$
moreover have $\left(\left(\lambda i \in I . \mathbf{1}_{G s}\right)\left(i:=e i, j:=\mathbf{1}_{G s} j\right)\right)$

$$
=\left(\left(\lambda i \in \text { insert } j I . \mathbf{1}_{G s}\right)(i:=e i)\right) \text { using } i j(2) \text { by auto }
$$

ultimately show ?thesis using $i j(2,4,6)$ unfolding $g$ by auto qed
ultimately show ?thesis by simp
qed
moreover have $x\left(\left(\lambda i \in(\right.\right.$ insert $\left.\left.j I) . \mathbf{1}_{G s} i\right)(j:=e j)\right)=g($ insert $j I) x j(e$
j)
unfolding $g$ by simp
ultimately show ?thesis by argo

## qed

finally show ?case using $j$ unfolding $g$ by auto
qed
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
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\congG group (DirProds Gs I) finite I
    shows DirProds (Characters ○ 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 ○ Gs) I
            using DirProds-finite-comm-group-iff[OF assms(3), of Characters ○ 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: \i. i\inI\Longrightarrowgroup (Gs i) using DirProds-group-imp-groups[OF assms(2)]
    let ?f = (\lambdacp. (\lambdae. (if e\incarrier (DirProds Gs I) then \i\inI.cp i (e i) else 0)))
    have f-in:?f x carrier (Characters (DirProds Gs I))
    if x:x\incarrier (DirProds (Characters ○Gs)I) for x
    proof(unfold carrier-Characters characters-def, safe, unfold-locales)
    show ?f x 1 1 DirProds Gs I }=
    proof -
```

```
    have \(x i\left(\mathbf{1}_{\text {DirProds }}\right.\) Gs \(\left.I i\right) \neq 0\) if \(i: i \in I\) for \(i\)
    proof -
        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 ix \(i\)
                using \(i x\) unfolding DirProds-def Characters-def characters-def by auto
            show ?thesis using DirProds-one' \([O F i\), of Gs] by simp
    qed
    thus ?thesis by (simp add: assms(3))
    qed
    show ?f \(x a=0\) if \(a \notin\) carrier (DirProds Gs \(I\) ) for \(a\) using that by simp
    show ?f \(x\left(a \otimes_{\text {DirProds Gs } I} b\right)=\) ?f \(x a *\) ?f \(x b\)
    if \(a b: a \in \operatorname{carrier}\) (DirProds Gs \(I\) ) \(b \in\) carrier (DirProds Gs \(I\) ) for \(a b\)
    proof -
    have \(a \otimes_{\text {DirProds }}\) Gs \(I \quad b \in\) carrier (DirProds Gs I) using that by blast
    moreover have \(\left(\prod i \in I . x i\left(\left(a \otimes_{\text {DirProds Gs } I} b\right) i\right)\right)\)
                    \(=\left(\prod i \in I . x i(a i)\right) *\left(\prod i \in I . x i(b i)\right)\)
    proof -
    have \(x i\left(\left(a \otimes_{\text {DirProds Gs I }}{ }^{b}\right) i\right)=x i(a i) * x i(b i)\) if \(i: i \in I\) for \(i\)
    proof -
            interpret xi: character Gs ixi
            using ix unfolding DirProds-def Characters-def characters-def by auto
                    show ?thesis using ab comp-mult \([O F\), of Gs a b] by (auto simp:
comp-in-carr \([O F-i])\)
            qed
            thus ?thesis using prod.distrib by force
        qed
        ultimately show ?thesis using that by auto
    qed
qed
have ?f \(\in\) iso (DirProds (Characters o 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 \(\circ G s\) ) \(I\) ) for \(x\) using \(f\)-in[OF that] .
    show ?f \(\left(x \otimes_{\text {DirProds }}\right.\) (Characters \(\left.\left.\circ G s\right) I y\right)=\) ?f \(x \otimes_{\text {Characters }}\) (DirProds Gs I)
?f \(y\)
            if \(x \in\) carrier (DirProds (Characters \(\circ\) Gs) I) \(y \in\) carrier (DirProds
(Characters \(\circ\) Gs) \(I\) )
            for \(x y\)
            proof -
                            have ?f \(x \otimes\) Characters (DirProds Gs I) ?f \(y\)
                            \(=\left(\lambda e\right.\). if \(e \in\) carrier (DirProds Gs I) then \(\left(\prod i \in I . x i(e i)\right) *\left(\prod i \in I . y i\right.\)
(e i)) else 0)
            unfolding Characters-def by auto
    also have \(\ldots=\) ?f \(\left(x \otimes_{\text {DirProds }}(\right.\) Characters \(\left.\circ G s) I y\right)\)
    proof -
```

```
                have (\prodi\inI. x i (e i))*(\prodi\inI. y i (e i))
                    =(\prodi\inI.(x \otimes DirProds (Characters ○Gs)Iy) i(ei)) 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 ○ Gs) I Characters
(DirProds Gs I) ?f
        by (unfold-locales, simp)
    show bij-betw ?f (carrier (DirProds (Characters ○ Gs) I)) (carrier (Characters
(DirProds Gs I)))
    proof (intro bij-betwI)
        let ?g = (\lambdac. (\lambdai\inI. (\lambdaa.c ((\lambdai\inI. 1 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\inI for x i
            using DirProds-subchar[OF DP.finite-comm-group-axioms x i assms(3)] .
            have g-in:?g x carrier (DirProds (Characters ○Gs)I)
            if x: x 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 ○ Gs) I) -> carrier (Characters
(DirProds Gs I))
            using f-in by fast
            show gi: ?g \in carrier (Characters (DirProds Gs I)) }->\mathrm{ carrier (DirProds
(Characters ○ Gs) I)
            using g-in by fast
            show ?f (?g x) =x if x: x 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 (\prodi\inI. (\lambdai\inI. \lambdaa.x x ((\lambdai\inI. 1 Gs i)(i:=a))) i (ei))=xe
            if e: e\incarrier (DirProds Gs I) for e
    proof -
            define y where y:y=(\lambdae. if e carrier (DirProds Gs I)
                            then \prodi\inI. (\lambdai\inI. \lambdaa.x ((\lambdai\inI. 1 Gs i})(i:=a))) 
(ei)
                        else 0)
        from Characters-DirProds-single-prod[OF DP.finite-comm-group-axioms x
assms(3)]
        have }y=x\mathrm{ using }y\mathrm{ by force
        hence y e=x e by blast
        thus ?thesis using e unfolding y by argo
        qed
        with x.char-eq-0 show ?thesis by force
```

```
    qed
    show ?g (?f x) =x if x: x\in carrier (DirProds (Characters ○Gs) I) for x
    proof(intro eq-parts-imp-eq[OF g-in[OF f-in[OF x]] x])
    show ?g (?f x) i=x i if i: i\inI for i
    proof -
        interpret xi: character Gs i x i
            using x i unfolding DirProds-def Characters-def characters-def by auto
        have ?g(?f x) i a = x i a if a: a\not\incarrier (Gs i) for a
        proof -
            have (\lambdai\inI. 1 (Gs i)(i:=a)\not\incarrier (DirProds Gs I)
                using a i unfolding DirProds-def PiE-def Pi-def by auto
            with xi.char-eq-O[OF a] a i show ?thesis by auto
        qed
        moreover have ?g(?f x) i a =x i a if a: a\incarrier (Gs i) for a
        proof -
            have (\lambdai\inI. 1 1 Gs i)}(i:=a)\in\operatorname{carrier (DirProds Gs I)
            using a i monoid.one-closed[OF group.is-monoid[OF allg]]
            unfolding DirProds-def by force
            moreover have (\prodj\inI.x j (((\lambdai\inI. 1 Gs i)(i:=a)) j))=x = i a
            proof -
                have (\prodj\inI. x j (((\lambdai\inI. 1 1 Gs i) (i:=a)) j))
                    =xi(((\lambdai\inI. 1 1 Gs i)(i:=a)) i)* (\prodj\inI-{i}.xj (((\lambdai\inI. 1 1 Gsi})(
:=a)(j))
                by (meson assms(3) i prod.remove)
            moreover have x j (((\lambdai\inI. 1 Gs i)(i:=a)) j)=1 if j: j\inIj\not=i for j
                proof -
                    interpret xj: character Gs j x j
                        using j(1)x unfolding DirProds-def Characters-def characters-def
by auto
                    show ?thesis using j by auto
                        qed
                    moreover have x i (((\lambdai\inI. 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 ○ 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 \(n s: \operatorname{DirProds}(\lambda n . Z(n s!n))\{. .<\) length \(n s\} \cong G \forall n \in\) set \(n s . n \neq 0\).
    interpret DP: group DirProds \((\lambda n . Z(n s!n))\{. .<\) length \(n s\}\)
        by (intro DirProds-is-group, auto)
    have \(G \cong \operatorname{DirProds}(\lambda n . Z(n s!n))\{. .<\) length ns \(\}\) using \(\operatorname{DP.iso-sym[OFns(1)]}\)
    moreover have DirProds (Characters \(\circ(\lambda n . Z(n s!n)))\{. .<\) length \(n s\} \cong\)
Characters \(G\)
    by (intro Characters-DirProds-iso[OF ns(1) DirProds-is-group], auto)
    moreover have DirProds \((\lambda n . Z(n s!n))\{. .<\) length \(n s\}\)
                \(\cong\) DirProds (Characters \(\circ(\lambda n . Z(n s!n)))\{. .<\) length \(n s\}\)
    proof (intro DirProds-iso1)
        fix \(i\) assume \(i: i \in\{. .<\) length \(n s\}\)
        obtain \(a\) where cyclic-group ( \(Z(n s!i)\) ) a using \(Z n\)-cyclic-group .
        then interpret Zi: cyclic-group \(Z(n s!i) a\).
        interpret Zi: finite-cyclic-group Z (ns!i) a
        proof
            have \(\operatorname{order}(Z(n s!i)) \neq 0\) using ns(2) \(i Z n\)-order by simp
        thus finite (carrier ( \(Z(n s!i))\) unfolding order-def by (simp add: card-eq-O-iff)
        qed
        show Group.group \(((\) Characters \(\circ(\lambda n . Z(n s!n)))\) i)
            Group.group \((Z(n s!i)) Z(n s!i) \cong(\) Characters \(\circ(\lambda n . Z(n s!n))) i\)
            using Zi.Characters-iso Zi.finite-comm-group-Characters comm-group-def fi-
nite-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 $G$
using iso-same-card [OF Characters-iso] unfolding order-def by argo
corollary (in finite-comm-group) card-characters: card (characters $G$ ) $=$ order $G$
using 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 $H$ G
shows $(\lambda \chi x$. if $x \in$ carrier $G$ then $\chi(H \#>x)$ else 0$) \in$

$$
\text { iso }(\text { Characters }(G \text { Mod } H))((\text { Characters } G) \| \text { carrier }:=\{\chi \in \text { characters }
$$ G. $\forall x \in H . \chi x=1\}$ D) 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 $C::\left({ }^{\prime} a \Rightarrow\right.$ 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)
next
case 2
thus ?case
by (auto simp: C-def one-Characters mult-Characters carrier-Characters char-
acters-def)
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 $f::\left({ }^{\prime} a\right.$ set $\Rightarrow$ complex $) \Rightarrow\left({ }^{\prime} a \Rightarrow\right.$ 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 $(G \operatorname{Mod} H) \chi$ 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)
    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)\mp@subsup{\otimes}{GModH}{M}(H#>y)
        using 3 by (intro char-mult [symmetric]) (auto simp: carrier-FactGroup)
    also have }(H#>x)\otimes\mp@subsup{\otimes}{G Mod H}{M}(H#>y)=H#>(x\otimesy
        using 3 by (simp add: H.rcos-sum)
    finally show ?case
        using 3 by (simp add: f-def)
    qed
qed
have [intro]: f \chi \inC 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\inH\mathrm{ for }
    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\mathrm{ iso (Characters (G Mod H)) ((Characters G)\ carrier := C\)
proof (rule isoI)
    show f G 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)
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 \chi1 \chi2)
            interpret \chi1: character G Mod H \chi1
                using 1 by (auto simp: characters-def)
            interpret \chi2: character G Mod H \chi2
                using 1 by (auto simp: characters-def)
            have \chi1 H'}=\chi2\mp@subsup{H}{}{\prime}\mathrm{ for }\mp@subsup{H}{}{\prime
            proof (cases H'}\mp@subsup{H}{}{\prime
```

```
        case False
        thus ?thesis by (simp add: \chi1.char-eq-0 \chi2.char-eq-0)
    next
    case True
    then obtain x where x:x\in carrier G H'}=H##>
        by (auto simp: carrier-FactGroup)
    from 1 have f \chi1 x = f \chi2 x
        by simp
    with x show ?thesis
        by (auto simp: f-def)
qed
thus \chi1 = \chi2 by force
qed
have f ' characters (G Mod H)\subseteqC
by (auto simp: characters-def)
moreover have C\subseteqf'characters ( G Mod H)
proof safe
fix }\chi\mathrm{ assume }\chi:\chi\in
from \chi interpret character G \chi
by (auto simp:C-def characters-def)
have [simp]: \chi x=1 if }x\inH\mathrm{ for }
    using }\chi\mathrm{ that by (auto simp: C-def)
    have \forallH'\incarrier (G Mod H). \existsx\incarrier G. H' }=H|#>
    by (auto simp: carrier-FactGroup)
    then obtain h where h: h H'\in carrier G H'}=H#>h\mp@subsup{H}{}{\prime}\mathrm{ if }\mp@subsup{H}{}{\prime}\in\mathrm{ carrier
(G Mod H) for H'
            by metis
        define }\mp@subsup{\chi}{}{\prime}\mathrm{ where }\mp@subsup{\chi}{}{\prime}=(\lambda\mp@subsup{H}{}{\prime}\mathrm{ . if }\mp@subsup{H}{}{\prime}\in\operatorname{carrier}(G\mathrm{ Mod H) then }\chi(h\mp@subsup{H}{}{\prime})\mathrm{ else
    have \chi-cong: \chi x = \chi y if H#> x=H#> y x\in carrier G y f carrier
    proof -
        have }x\inH#>>
        by (simp add: H.subgroup-axioms rcos-self that(2))
        also have ... = H #> y
        by fact
        finally obtain z where z:z\inHx=z\otimesy
        unfolding r-coset-def by auto
        thus ?thesis
        using z H.subset that by simp
    qed
    have character (G Mod H) \chi}\mp@subsup{\chi}{}{\prime
    proof (unfold-locales, goal-cases)
        case 1
        have H:H\in\operatorname{carrier (G Mod H)}
```

0) 

$G$ for $x y$
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 $\chi^{\prime}$-def)
next
case ( $2 H^{\prime}$ )
thus ?case by (auto simp: $\chi^{\prime}$-def)
next
case (3 H1 H2)
from 3 have H12: H1 < \#> H2 $\in \operatorname{carrier}(G \operatorname{Mod} H)$
using Fact.m-closed by force
have $\chi(h(H 1<\#>H 2))=\chi(h H 1 \otimes h H 2)$
proof (rule $\chi$-cong)
show $H \#>h(H 1<\#>H 2)=H \#>(h H 1 \otimes h H 2)$
by (metis 3 H.rcos-sum H12 h)
qed (use 3 h[of H1] $h[$ of H2] $h[O F$ H12] in auto)
thus? case
using 3 H12 $h\left[\right.$ of H1] $h\left[\right.$ of H2] by (auto simp: $\chi^{\prime}$-def)
qed
moreover have $f \chi^{\prime} x=\chi x$ for $x$
proof (cases $x \in$ carrier $G$ )
case False
thus ?thesis
by (auto simp: $f$-def $\chi^{\prime}$-def char-eq- 0 -iff)
next
case True
hence $*: H \#>x \in \operatorname{carrier}(G \operatorname{Mod} H)$
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 $\chi^{\prime}$-def)
qed
hence $f \chi^{\prime}=\chi$ by force
ultimately show $\chi \in f$ 'characters ( $G$ Mod $H$ )
unfolding characters-def by blast
qed
ultimately show $f$ 'characters $(G \operatorname{Mod} H)=C$
by blast
qed
thus bij-betw $f($ carrier (Characters $(G$ Mod $H))$ ) (carrier (Characters $G \$ carrier $:=C(D))$
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 $\Rightarrow\left({ }^{\prime} a \Rightarrow\right.$ complex $) \Rightarrow\left({ }^{\prime} a \Rightarrow\right.$ 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 $:=H D)$ ) (restrict-char H)
proof -
let ? $C G=$ Characters $G$
let $? H=G($ 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 $C G$ : finite-comm-group? $C G$ 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 ? $C H$ if $x: x \in$ carrier ? $C G$ for $x$
proof -
interpret $x c$ : character $G x$ using $x$ unfolding Characters-def characters-def
by $\operatorname{simp}$
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 $H\left(x \otimes_{\text {? } C G} y\right)=$ restrict-char $H x \otimes_{\text {? } C H}$ restrict-char $H y$
if $x: x \in$ carrier ? $C G$ and $y: y \in$ carrier ? $C G$ for $x y$
proof -
interpret $x$ : character $G x$ using $x$ unfolding Characters-def characters-def by $\operatorname{simp}$
interpret $y c:$ character $G y$ using $y$ unfolding Characters-def characters-def by $\operatorname{simp}$
show ?thesis unfolding Characters-def restrict-char-def by auto
qed
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\carrier :=H|)) (restrict-char H)
        = {\chi\incharacters G. }\forallx\inH.\chix=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 $H G$
shows restrict-char $H^{\prime}($ carrier $($ Characters $G))=$ carrier $($ Characters $(G \$ carrier := $H$ D))
proof -
interpret $H$ : subgroup $H G$ by fact
interpret $H$ : finite-comm-group $G($ carrier $:=H)$ using subgroup-imp-finite-comm-group $[O F$
assms] .
interpret $r$ : group-hom Characters $G$ Characters $(G \$ carrier $:=H \mid)$ restrict-char H using restrict-char-hom[OF assms].
interpret Mod: finite-comm-group G Mod H using finite-comm-FactGroup[OF assms] .
interpret $C G$ : 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 $H * \operatorname{card}($ kernel (Characters $G)($ Characters $(G($ carrier $:=H \mid)))$ (restrict-char $H$ )) $=$ order $G$
using 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 $(G(\operatorname{carrier}:=H \mid)))$
(restrict-char $H)$ ) $=0$
using r.one-in-kernel unfolding kernel-def CG.fin by auto
ultimately have c2: card $H=$ card (restrict-char $H^{\prime}$ carrier (Characters $G$ ))
using r.image-kernel-product[unfolded order-Characters] by (metis mult-right-cancel)
have restrict-char $H^{\prime}($ carrier $($ Characters $G)) \subseteq$ carrier $($ Characters $(G \backslash$ carrier := $H$ D))
by auto
with $c 2$ 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(|arrier :=H|)) \chi
    obtains }\mp@subsup{\chi}{}{\prime}\mathrm{ where character }G\mp@subsup{\chi}{}{\prime}\mathrm{ and }\bigwedgex.x\inH\Longrightarrow\mp@subsup{\chi}{}{\prime}x=\chi
proof -
    from restrict-char-image[OF assms(1)] assms(2) obtain }\mp@subsup{\chi}{}{\prime
        where chi': restrict-char H \chi}\mp@subsup{}{\prime}{\prime}=\chi\mathrm{ character G }\mp@subsup{\chi}{}{\prime
        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 $H$ that share the values with them is the same for both.

```
lemma (in finite-comm-group) character-restrict-card:
    assumes subgroup \(H\) character \(G\) a character \(G b\)
    shows card \(\left\{\chi^{\prime} \in\right.\) characters \(\left.G . \forall x \in H . \chi^{\prime} x=a x\right\}=\operatorname{card}\left\{\chi^{\prime} \in\right.\) characters \(G\).
\(\left.\forall x \in H . \chi^{\prime} x=b x\right\}\)
proof -
    interpret \(H\) : subgroup \(H G\) by fact
    interpret \(H\) : finite-comm-group \(G(\) carrier \(:=H D\) using \(\operatorname{assms}(1)\)
        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 \(b c: b \in\) carrier (Characters \(G\) ) unfolding Characters-def characters-def
using assms by simp
    define \(f\) where \(f: f=\left(\lambda c . b \otimes_{\text {Characters } G}{ }^{\text {inv }}\right.\) Characters \(\left.G{ }^{a} \otimes_{\text {Characters } G} c\right)\)
    define \(g\) where \(g: g=\left(\lambda c . a \otimes_{\text {Characters } G}{ }^{\text {inv }}\right.\) Characters \(G{ }^{b} \otimes_{\text {Characters }} G\) c)
    let \(? A=\left\{\chi^{\prime} \in\right.\) characters \(\left.G . \forall x \in H . \chi^{\prime} x=a x\right\}\)
    let ? \(B=\left\{\chi^{\prime} \in\right.\) characters \(\left.G . \forall x \in H . \chi^{\prime} x=b x\right\}\)
    have bij-betw \(f\) ? A ? B
    proof(intro bij-betwI[of -- -g])
        show \(f \in ? A \rightarrow\) ? \(B\)
        proof
            show \(f x \in ? B\) if \(x: x \in ? A\) for \(x\)
            proof -
                interpret \(x c\) : 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 \(y\)
            proof -
                have \(\left({ }^{\text {inv }}\right.\) Characters \(\left.G a\right) y * a y=1\)
                    by (simp add: a.inv-Characters a.mult-inv-character mult.commute
principal-char-def \(y\) )
                    thus ?thesis unfolding \(f\) mult-Characters using \(x y\) by fastforce
                    qed
```

thus $f x \in$ ?B unfolding $f$ carrier-Characters[symmetric] using ac bc xc by blast
qed
qed
show $g \in ? B \rightarrow$ ? $A$
proof
show $g x \in ? A$ if $x: x \in ? B$ for $x$
proof -
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 $y$
proof -
have $\left({ }^{\text {inv }}\right.$ Characters $\left.G b\right) y * x y=1$ using $x y$
by (simp add: b.inv-Characters b.mult-inv-character mult.commute principal-char-def)
thus ?thesis unfolding $g$ 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 $x$
proof -
have $x c: x \in$ carrier (Characters $G$ ) using $x$ unfolding Characters-def by force
with $a c b c$ 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 $x$
proof -
have $x c: x \in$ 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|) \chi
    shows card {\chi'\incharacters G.}\forallx\inH. \chi' x=\chix}* card H=order G
proof -
    interpret H: subgroup H G by fact
```

```
    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 < carrier (Characters G) restrict-char H a=\chi
    proof -
        have \existsa\incarrier (Characters G). restrict-char H a = \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\incharacters G. }\forallx\inH.\chix=1}={\chi\incharacters G. \forallx\inH.\chi
= principal-char G x}
            unfolding principal-char-def by force
    have ac: {\chi}\mp@subsup{\chi}{}{\prime}\in\mathrm{ characters G. }\forallx\inH.\mp@subsup{\chi}{}{\prime}x=\chix}={\mp@subsup{\chi}{}{\prime}\in\mathrm{ characters G. }\forallx\inH
\chi}\mp@subsup{}{}{\prime}x=ax
            using a(2) unfolding restrict-char-def by force
    have card {\chi\incharacters G. }\forallx\inH.\chix=1}=card {\chi'\incharacters G. \forallx\inH
\chi'}x=\chix
            by (unfold ac p; intro character-restrict-card[OF assms(1)],
                use a[unfolded Characters-def characters-def] in auto)
    moreover have card {\chi\incharacters G. }\forallx\inH.\chix=1}=\operatorname{card}(\operatorname{carrier}(
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 G
            using 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 \(\chi\) on \(G\) such that \(\chi(x)=z\).
```

```
lemma (in group) powi-get-exp-self:
```

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

```
    finally show ?thesis.
```


## qed

corollary (in finite-comm-group) character-with-value-exists:
assumes $x \in$ carrier $G$ and $x \neq 1$ and $z^{\wedge}$ ord $x=1$
obtains $\chi$ where character $G \chi$ and $\chi x=z$
proof -
interpret $H$ : subgroup generate $G\{x\} G$ using generate-is-subgroup assms(1) by $\operatorname{simp}$
interpret $H$ : finite-comm-group $G($ carrier := generate $G\{x\} \mid)$
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 $G\{x\} \mid) x$
proof (intro H.cyclic-groupIO)
show $x \in$ carrier $(G \backslash$ carrier $:=$ generate $G\{x\} \emptyset)$ using generate.incl $[$ of $x$ $\{x\}$ G] by simp
show carrier $(G($ carrier $:=$ generate $G\{x\} \emptyset)=$ generate $(G \|$ carrier $:=$ generate $G\{x\} D)\{x\}$
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 $(G \mid$ carrier := generate $G\{x\} \mid) c$
$c=(\lambda a$. if $a \in$ carrier $(G \mid$ carrier $:=$ generate $G\{x\} \mid))$
then $z$ powi H.get-exp $x$ a else 0) by blast
have $c x: c x=z$ 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 $[\operatorname{OF} f(1)]$, use $c x f(2)$ generate-sincl in blast)
qed
In particular, for any $x$ that is not the identity element, there exists a character $\chi$ such that $\chi(x) \neq 1$.
corollary (in finite-comm-group) character-neq-1-exists:
assumes $x \in$ carrier $G$ and $x \neq 1$
obtains $\chi$ where character $G \chi$ and $\chi x \neq 1$
proof -
define $z$ where $z=\operatorname{cis}(2 *$ pi / ord $x)$
have $z$-pow-h: $z^{\wedge}$ ord $x=1$
by (auto simp: $z$-def DeMoivre)
from assms have ord $x \geq 1$ 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 \([\operatorname{simp}]: z \neq 1\)
proof
    assume \(z=1\)
    have bij-betw \((\lambda k\). cis \((2 *\) pi \(*\) real \(k / \operatorname{real}(\) ord \(x)))\{. .<\) ord \(x\}\left\{z . z^{\wedge}\right.\) ord \(x\)
\(=1\}\)
    using \(\langle\) ord \(x>1\) 〉 by (intro bij-betw-roots-unity) auto
    hence \(\operatorname{inj}\) : inj-on \((\lambda k\).cis \((2 * p i *\) real \(k / \operatorname{real}(\operatorname{ord} x)))\{. .<\) ord \(x\}\)
            by (auto simp: bij-betw-def)
    have \(0=(1::\) nat \()\)
        using \(\langle z=1\rangle\) and \(\langle o r d x>1\rangle\) by (intro inj-onD[OF inj]) (auto simp: \(z\)-def)
    thus False by simp
qed
obtain \(\chi\) where character \(G \chi\) and \(\chi x=z\)
    using character-with-value-exists[OF assms \(z\)-pow-h].
    thus ?thesis using that [of \(\chi]\) by simp
qed
```


### 1.6 The first orthogonality relation

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

```
corollary (in finite-comm-group) character-orthogonality1:
    assumes character \(G \chi\) and character \(G \chi^{\prime}\)
    shows \(\left(\sum x \in\right.\) carrier \(\left.G . \chi x * \operatorname{cnj}\left(\chi^{\prime} x\right)\right)=\left(\right.\) if \(\chi=\chi^{\prime}\) then of-nat (order \(\left.G\right)\)
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 \(\chi^{\prime} x\)
    interpret character \(G \lambda x . \chi x *\) inv-character \(\chi^{\prime} x\)
        by (intro character-mult character.inv-character assms)
    have \(\left(\sum x \in\right.\) carrier \(G\). \(\left.\chi x * \operatorname{cnj}\left(\chi^{\prime} x\right)\right)=\left(\sum x \in\right.\) carrier \(G\). ? \(\left.\chi x\right)\)
    by (intro sum.cong) (auto simp: inv-character-def)
    also have \(\ldots=(\) if \(? \chi=\) principal-char \(G\) then of-nat \((\) order \(G)\) else 0\()\)
        by (rule sum-character)
    also have ? \(\chi=\) principal-char \(G \longleftrightarrow \chi \otimes_{C}{ }^{\text {inv }}{ }_{C} \chi^{\prime}=\mathbf{1}_{C}\)
    using assms by (simp add: Characters-simps characters-def)
    also have \(\ldots \longleftrightarrow \chi=\chi^{\prime}\)
    proof
        assume \(\chi \otimes_{C}{ }^{i n v}{ }_{C} \chi^{\prime}=\mathbf{1}_{C}\)
        from C.inv-equality [OF this] and assms show \(\chi=\chi^{\prime}\)
            by (auto simp: characters-def Characters-simps)
    next
        assume \(*: \chi=\chi^{\prime}\)
    from assms show \(\chi \otimes_{C}{ }^{\text {inv }}{ }_{C} \chi^{\prime}=\mathbf{1}_{C}\)
        by (subst *, intro C.r-inv) (auto simp: carrier-Characters characters-def)
    qed
    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 $|\widehat{\widehat{G}}|=|\widehat{G}|=|G|$ 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 $x=(\lambda \chi$. if character $G \chi$ then $\chi x$ 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 Char-
acters-def
                                principal-char-def character.char-eq-0)
qed
lemma double-dual-iso-mult [simp]:
    assumes x c carrier G y \in carrier G
    shows double-dual-iso (x\otimesy)=
                double-dual-iso x © 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 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=1
        if x:x\in carrier G double-dual-iso x = 1 Characters (Characters G) for }
    proof (rule ccontr)
        assume }x\not=
        obtain }\chi\mathrm{ where }\chi\mathrm{ : character }G\chi\chix\not=
            using character-neq-1-exists[OF x(1)<x\not=1\rangle].
        from x have }\forall\chi.(\mathrm{ if }\chi\in\mathrm{ characters G then }\chix\mathrm{ else 0) }=(\mathrm{ if }\chi\in\mathrm{ characters
G then 1 else 0)
        by (auto simp: double-dual-iso-def Characters-def fun-eq-iff
                                    principal-char-def characters-def)
    hence eq1: }\forall\chi\in\mathrm{ characters G. }\chix=1 by meti
    with \chi 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\Longrightarrowy\in\mathrm{ carrier }G\Longrightarrow\mathrm{ double-dual-iso }x=\mathrm{ double-dual-iso }y
x=y
```

```
    by (auto dest: inj-onD[OF inj-double-dual-iso])
theorem double-dual-iso:double-dual-iso }\in\mathrm{ 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)
    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)
    next
        have card (carrier (Characters (Characters G))) = card (carrier G)
                            by (simp add: carrier-Characters G'.card-characters card-characters or-
der-def)
            also have ... = 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)
    qed
    qed
qed
lemma double-dual-is-iso: Characters (Characters G)\congG
    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 $\left(\sum \chi \in\right.$ characters $\left.G . \chi x\right)=($ if $x=\mathbf{1}$ then of-nat $($ order $G)$ else 0$)$
proof -
interpret $G^{\prime}$ : finite-comm-group Characters $G$
by (rule finite-comm-group-Characters)
interpret $x$ : character Characters $G$ double-dual-iso $x$
using $x$ by auto

```
    from x.sum-character show ?thesis using double-dual-iso-eq-iff[of x 1] x
    by (auto simp: characters-def carrier-Characters order-Characters simp del:
double-dual-iso-eq-iff)
qed
corollary character-orthogonality2:
    assumes x c carrier G y \in carrier G
    shows (\sum\chi\incharacters G. \chi x*cnj (\chi y)) =(if x=y then of-nat (order G)
else 0)
proof -
    from assms have (\sum\chi\incharacters G. \chi x* cnj (\chi y)) = (\sum\chi\incharacters G.
\chi(x\otimesinv y))
    by (intro sum.cong) (simp-all add: character.char-inv character.char-mult char-
acters-def)
    also from assms have ... = (if x \otimesinv y = 1 then of-nat (order G) else 0)
            by (intro sum-characters) auto
    also from assms have }x\otimes\mathrm{ inv y=1 u
            using inv-equality[of x inv y] by auto
    finally show ?thesis .
qed
end
```

no-notation integer-mod-group $(Z)$
end

## 2 Dirichlet Characters

theory Dirichlet-Characters<br>imports<br>Multiplicative-Characters<br>HOL-Number-Theory.Residues<br>Dirichlet-Series.Multiplicative-Function<br>begin

Dirichlet characters are essentially just the characters of the multiplicative group of integer residues $\mathbb{Z} \mathbb{Z} / n \mathbb{Z} \mathbb{Z}$ 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 $\chi(k$ $\bmod n)=\chi(k)$ 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 $n=1$ carrier $=$ totatives $n$, monoid.mult $=(\lambda x y .(x * y)$ $\bmod n)$, one $=1$ )

```
definition principal-dchar :: nat }=>\mathrm{ nat }=>\mathrm{ complex where
    principal-dchar n}=(\lambdak\mathrm{ . if coprime k n then 1 else 0)
lemma principal-dchar-coprime [simp]: coprime k n\Longrightarrow principal-dchar n k = 1
    and principal-dchar-not-coprime [simp]: ᄀcoprime 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>0
    shows principal-dchar n (n-Suc 0) =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 בa mod n t totatives n \longleftrightarrow 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 (\chi\mp@subsup{0}{}{1})
end
locale residues-nat =
    fixes n :: nat (structure) and G
    assumes n: n>1
    defines }G\equiv\mathrm{ residue-mult-group n
begin
lemma order [simp]: order G = totient n
    by (simp add: order-def G-def totient-def residue-mult-group-def)
lemma totatives-mod [simp]: x totatives }n\Longrightarrowx\operatorname{mod}n=
    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 f carrier G
    hence coprime (x*y) n
```

```
    by (auto simp: G-def residue-mult-group-def totatives-def)
    with }xy\mathrm{ and n show }x\mp@subsup{\otimes}{G}{}y\in\mathrm{ carrier }
    using coprime-common-divisor-nat[of x*yn]
    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)
next
    fix x y z assume xyz:x\in carrier G y farrier Gz\in carrier G
    thus }x\mp@subsup{\otimes}{G}{}y\mp@subsup{\otimes}{G}{}z=x\mp@subsup{\otimes}{G}{}(y\mp@subsup{\otimes}{G}{}z
        by (auto simp: G-def residue-mult-group-def mult-ac mod-mult-right-eq)
next
    fix x assume }x\in\mathrm{ carrier }
    with n have x<n by (auto simp: G-def residue-mult-group-def totatives-def
                intro!: order.not-eq-order-implies-strict)
    thus }\mp@subsup{\mathbf{1}}{G}{}\mp@subsup{\otimes}{G}{}x=x\mathrm{ and }x\mp@subsup{\otimes}{G}{}\mp@subsup{\mathbf{1}}{G}{}=
    by (simp-all add:G-def residue-mult-group-def)
next
    have x\inUnits G if x 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] (mod n)
        by (subst (asm) coprime-iff-invertible'-nat) auto
    hence }x*y\operatorname{mod}n=
                using n by (simp add: cong-def mult-ac)
    moreover from }y\mathrm{ have coprime y n
        by (subst coprime-iff-invertible-nat) (auto simp: mult.commute)
    ultimately show \existsa\incarrier G. a \otimes G
            by (intro bexI[of - y])
                (auto simp: G-def residue-mult-group-def totatives-def mult.commute intro!:
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 \(c 2 d c::(\) nat \(\Rightarrow\) complex \() \Rightarrow(\) nat \(\Rightarrow\) complex \()\) where
```

    \(c 2 d c \chi=(\lambda x \cdot \chi(x \bmod n))\)
    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 $d c 2 c(c 2 d c \chi)=\chi$
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)
qed
end
locale dcharacter = residues-nat +
    fixes \chi :: nat }=>\mathrm{ complex
    assumes mult-aux: a totatives n\Longrightarrowb\in totatives n\Longrightarrow\chi(a*b)=\chia*\chi
b
    assumes eq-zero: \negcoprime a }n\Longrightarrow\chia=
    assumes periodic: }\chi(a+n)=\chi
    assumes one-not-zero: \chi 1}\not=
begin
lemma zero-eq-0 [simp]: \chi 0 = 0
    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)=\chia
proof (induction m)
    case (Suc m)
    have }a+Sucm*n=a+m*n+n by sim
    also have \chi\ldots=\chi(a+m*n) by (rule periodic)
    also have ... = \chi a by (rule Suc.IH)
    finally show ?case.
qed simp-all
lemma minus-one-periodic [simp]:
    assumes k>0
    shows }\quad\chi(k*n-1)=\chi(n-1
proof -
    have }k*n-1=n-1+(k-1)*
        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:
    assumes [a=b](mod n)
    shows }\quad\chia=\chi
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)=\chib\mathrm{ 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)=\chia*\chib
proof (cases coprime a n ^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\operatorname{mod}n)*(b\operatorname{mod}n))=\chi(a\operatorname{mod}n)*\chi(b\operatorname{mod}n
    by (rule mult-aux)
    also have \chi ((a mod n)*(b\operatorname{mod}n))=\chi(a*b)
    by (rule cong) (auto simp: cong-def mod-mult-eq)
    finally show ?thesis by simp
next
    case False
    hence \negcoprime ( }a*b)n\mathrm{ 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 }\longleftrightarrow\mathrm{ नcoprime }x\mathrm{ n
proof safe
    assume }\chix=0\mathrm{ and coprime }x
    from cong-solve-coprime-nat [OF this(2)]
        obtain }y\mathrm{ where [x*y=Suc 0] (mod n) by blast
    hence }\chi(x*y)=\chi(Suc 0) by (rule cong
    with }\langle\chix=0\rangle\mathrm{ show False by simp
qed (auto simp: eq-zero)
lemma minus-one': \chi (n-1)\in{-1,1}
proof -
    define }\mp@subsup{n}{}{\prime}\mathrm{ where }\mp@subsup{n}{}{\prime}=n-
    have n: n=Suc (Suc n') using n by (simp add: n'-def)
    have (n-1)^2 = 1 + (n-2)*n
    by (simp add: power2-eq-square algebra-simps n)
    also have \chi ...=1
    by (subst periodic-mult) auto
    also have \chi ((n-1) ^2) = \chi (n-1)^2
    by (rule mult.power)
    finally show ?thesis
    by (subst (asm) power2-eq-1-iff) auto
qed
```

```
lemma c2dc-dc2c [simp]: c2dc (dc2c \chi) = \chi
    using }n\mathrm{ by (auto simp: c2dc-def dc2c-def fun-eq-iff intro!: cong simp: cong-def)
lemma character-dc2c: character G (dc2c \chi)
    by standard (insert n, auto simp: G-def residue-mult-group-def dc2c-def tota-
tives-def
                    intro!: eq-zero)
sublocale dc2c: character G dc2c \chi
    by (fact character-dc2c)
lemma dcharacter-inv-character [intro]: dcharacter n (inv-character \chi)
    by standard (auto simp: inv-character-def eq-zero periodic)
lemma norm: norm ( }\chik)=(\mathrm{ if coprime }kn\mathrm{ 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\operatorname{mod}n)\mathrm{ by (simp add:dc2c-def)
    also from n have norm ... = (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 }=>\mathrm{ (nat }=>\mathrm{ complex) set where
    dcharacters }n={\chi.\mathrm{ dcharacter n }\chi
context residues-nat
begin
lemma character-dc2c:dcharacter n \chi \Longrightarrow character G (dc2c \chi)
    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\mathrm{ coprime x n
        thus c2dc \chi x = 0
            by (auto simp: c2dc-def char-eq-0-iff G-def residue-mult-group-def tota-
```

```
tives-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 n
    by (simp add: principal-dchar-altdef dcharacter-c2dc | 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 bij-betw-dcharacters-characters:
    bij-betw dc2c (dcharacters n) (characters G)
    by (intro bij-betwI[where ?g = c2dc])
        (auto simp: characters-def dcharacters-def dcharacter-c2dc
            character-dc2c dcharacter.c2dc-dc2c)
lemma bij-betw-characters-dcharacters:
    bij-betw c2dc (characters G) (dcharacters n)
    by (intro bij-betwI[where ?g = dc2c])
        (auto simp: characters-def dcharacters-def dcharacter-c2dc
                            character-dc2c dcharacter.c2dc-dc2c)
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 n
    using 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=\mathrm{ principal-dchar }n\longleftrightarrow\chi=\mathrm{ principal-dchar }
    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:
    (\sumx\intotatives n. \chi x) = (if \chi = principal-dchar n then of-nat (totient n) else
0)
proof -
    from n have (\sumx\intotatives n. \chi x) = (\sumx\incarrier G. dc2c \chi x)
    by (intro sum.cong) (auto simp: totatives-def dc2c-def G-def residue-mult-group-def)
    also have ... = (if dc2c \chi = principal-char G then of-nat (order G) else 0)
    by (rule dc2c.sum-character)
    also have dc2c \chi= principal-char }G\longleftrightarrow\chi=\mathrm{ principal-dchar n
        by (metis c2dc-dc2c dc2c-principal principal-dchar-altdef)
    finally show ?thesis by simp
qed
lemma (in dcharacter) sum-dcharacter-block:
    (\sumx<n.\chi x)=(if \chi = principal-dchar n then of-nat (totient n) else 0)
proof -
    from n have ( }\sumx<n.\chix)=(\sumx\in\mathrm{ totatives n. }\chix
        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 \chi = 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 = \lambdak. if k=n then 0 else k and ?g=\lambdak. if k=0 then n else k
    have sum }\chi{1..n}=sum \chi{..<n
        using n by (intro sum.reindex-bij-witness[where j=?f and i=?g]) (auto
simp: eq-zero-iff)
    thus ?thesis by (simp add: sum-dcharacter-block)
qed
lemma (in dcharacter) sum-lessThan-dcharacter:
    assumes }\chi\not=\mathrm{ principal-dchar n
    shows }(\sumx<m.\chix)=(\sumx<m\operatorname{mod}n.\chix
proof (induction m rule: less-induct)
    case (less m)
    show ?case
    proof (cases m<n)
        case True
        thus ?thesis by simp
    next
        case False
    hence {..<m}={..<n}\cup{n..<m} by auto
    also have (\sumx\in\ldots..\chix)=(\sumx<n. \chix) +( \sumx\in{n..<m}. \chi x)
            by (intro sum.union-disjoint) auto
```

also from assms have $\left(\sum x<n . \chi x\right)=0$
by (subst sum-dcharacter-block) simp-all
also from False have $\left(\sum x \in\{n . .<m\} . \chi x\right)=\left(\sum x \in\{. .<m-n\} . \chi(x+n)\right)$ by (intro sum.reindex-bij-witness $[$ of $-\lambda x . x+n \lambda x . x-n]$ ) (auto simp: periodic)
also have $\ldots=\left(\sum x \in\{. .<m-n\} . \chi x\right)$ by (simp add: periodic)
also have $\ldots=\left(\sum x<(m-n) \bmod n . \chi x\right)$
using False and $n$ by (intro less.IH) auto
also from False and $n$ have $(m-n) \bmod n=m \bmod n$ 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 $\left(\sum x<m . \chi x\right) \leq$ totient $n$
proof -
have $\left(\sum x<m . \chi x\right)=\left(\sum x<m \bmod n . \chi x\right)$ by (rule sum-lessThan-dcharacter $)$
fact
also have $\ldots=\left(\sum x \mid x<m\right.$ mod $n \wedge$ coprime $\left.x n . \chi x\right)$
by (intro sum.mono-neutral-right) (auto simp: eq-zero-iff)
also have norm $\ldots \leq\left(\sum x \mid x<m \bmod n \wedge\right.$ coprime $x$ n. 1)
by (rule sum-norm-le) (auto simp: norm)
also have $\ldots=\operatorname{card}\{x . x<m \bmod n \wedge$ 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 $x: x \in\{x . x<m \bmod n \wedge$ coprime $x n\}$
hence $x<m \bmod n$ by simp
also have $\ldots<n$ using $n$ by simp
finally show $x \in$ totatives $n$ using $x$
by (auto simp: totatives-def intro!: Nat.grOI)
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 $\left(\sum x \leq m . \chi x\right) \leq$ totient $n$
using sum-dcharacter-lessThan-le[OF assms, of Suc m] by (subst (asm) lessThan-Suc-atMost)
lemma (in residues-nat) sum-dcharacters:
$\left(\sum \chi \in d\right.$ characters $\left.n . \chi x\right)=($ if $[x=1](\bmod n)$ 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 $\left(\sum \chi \in d\right.$ characters $\left.n . \chi x\right)=\left(\sum \chi \in\right.$ characters $G$. $\left.c 2 d c \chi x\right)$
by (rule sum.reindex-bij-betw [OF bij-betw-characters-dcharacters, symmetric])

```
    also from x have ... =( \sum\chi\incharacters G. \chi (x mod n))
    by (simp add: c2dc-def)
    also from x have ... = (if x mod n=1 then order G else 0)
    by (subst sum-characters) (unfold G-def residue-mult-group-def, auto)
    also from n have }x\operatorname{mod}n=1\longleftrightarrow[x=1](\operatorname{mod}n
    by (simp add: cong-def)
    finally show ?thesis by simp
next
    case False
    have }x\operatorname{mod}n\not=
    proof
        assume *: x mod n=1
    have gcd ( }x\operatorname{mod}n)n=1\mathrm{ by (subst *) simp
    also have gcd (x mod n) n= gcd x n
        by (subst gcd.commute) (simp only: gcd-red-nat [symmetric])
    finally show False using «\negcoprime x n> unfolding coprime-iff-gcd-eq-1 by
contradiction
    qed
    from False have (\sum\chi\indcharacters n. \chi x)=0
    by (intro sum.neutral) (auto simp: dcharacters-def dcharacter.eq-zero)
    with «x mod n}=1\rangle\mathrm{ and n show ?thesis by (simp add: cong-def)
qed
lemma (in dcharacter) even-dcharacter-linear-sum-eq-0 [simp]:
    assumes }\chi\not=\mathrm{ principal-dchar }n\mathrm{ and }\chi(n-1)=
    shows (\sumk=Suc 0..<n. of-nat k*\chik)=0
proof -
    have (\sumk=1..<n. of-nat k* \chi k)=(\sumk=1..<n. (of-nat n - of-nat k)*\chi(n
-k))
    by (intro sum.reindex-bij-witness[where i=\lambdak.n - k and j=\lambdak.n-k])
        (auto simp: of-nat-diff)
    also have ... =n* (\sumk=1..<n. \chi (n-k)) - (\sumk=1..<n. k*\chi (n-k))
    by (simp add: algebra-simps sum-subtractf sum-distrib-left)
    also have (\sumk=1..<n. \chi (n-k))=(\sumk=1..<n. \chi k)
    by (intro sum.reindex-bij-witness[where i=\lambdak.n-k and j=\lambdak.n-k])
auto
    also have ... = (\sumk<n. \chi k)
        by (intro sum.mono-neutral-left) (auto simp: Suc-le-eq)
    also have ... = 0 using assms by (simp add: sum-dcharacter-block)
    also have (\sumk=1..<n.of-nat k*\chi (n-k))=(\sumk=1..<n.k*\chik)
    proof (intro sum.cong refl)
    fix }k\mathrm{ assume }k:k\in{1..<n
    have of-nat k*\chik=of-nat k*\chi((n-1)*k)
            using assms by (subst mult) simp-all
    also have (n-1)*k=n-k+(k-1)*n
            using }k\mathrm{ by (simp add: algebra-simps)
    also have \chi ...= \chi (n-k)
        by (rule periodic-mult)
    finally show of-nat k*\chi(n-k)=of-nat k*\chik..
```

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


## 3 Dirichlet $L$-functions

theory Dirichlet-L-Functions imports<br>Dirichlet-Characters<br>HOL-Library.Landau-Symbols<br>Zeta-Function.Zeta-Function<br>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 $\chi$.
First of all, we need to take care of a syntactical problem: The notation for vectors uses $\chi$ 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 $\zeta$ 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 \(m \chi s=\)
        (if \(s=1\) then
            if \(\chi=\) principal-dchar \(m\) then 0 else eval-fds \((f d s \chi) 1\)
        else
            of-nat \(m\) powr \(-s *\left(\sum k=1 . . m . \chi k *\right.\) hurwitz-zeta \((\) real \(k /\) real \(\left.\left.m) s\right)\right)\)
lemma Dirichlet-L-conv-hurwitz-zeta-nonprincipal:
    assumes \(s \neq 1\)
    shows Dirichlet-L \(n \chi s=\)
                of-nat \(n\) powr \(-s *\left(\sum k=1 . . n . \chi k *\right.\) hurwitz-zeta \((\) real \(k /\) real \(\left.n) s\right)\)
    using assms by (simp add: Dirichlet-L-def)
```

Analyticity everywhere except 1 is trivial by the above definition, since the Hurwitz $\zeta$ 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>01\not\inA
    shows Dirichlet-L m \chi holomorphic-on A
proof -
    have (\lambdas. of-nat m powr - s*( }\sumk=1..m. \chi k*hurwitz-zeta (real k / real
m) s))
                holomorphic-on A
        using assms unfolding Dirichlet-L-def by (intro holomorphic-intros) auto
    also have ?this \longleftrightarrow ?thesis
        using assms by (intro holomorphic-cong refl) (auto simp: Dirichlet-L-def)
    finally show ?thesis.
qed
```


## context dcharacter <br> begin

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

```
lemma
    fixes s :: complex
    assumes s:Re s>1
    shows abs-summable-Dirichlet-L: summable (\lambdan.norm ( }\chin*\mathrm{ of-nat n powr
-s))
    and summable-Dirichlet-L: summable ( }\lambdan.\chin*\mathrm{ of-nat n powr -s)
    and sums-Dirichlet-L: (\lambdan.\chi n*n powr -s) sums Dirichlet-L n \chi s
    and Dirichlet-L-conv-eval-fds-weak: Dirichlet-L n \chi s = eval-fds (fds \chi)s
proof -
    define L where L}=(\sumn.\chin* of-nat n powr -s
    show summable ( }\lambdan\mathrm{ . norm ( }\chin*\mathrm{ 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 ( }\lambdan.\chin*\mathrm{ of-nat n powr -s)
        by (rule summable-norm-cancel)
    hence ( }\lambdan.\chin* of-nat n powr -s) sums L by (simp add: L-def sums-iff
    from this have (\lambdam. \sumk=m*n..<m*n+n. \chik*of-nat k powr - s) sums
L
    by (rule sums-group) (use n in auto)
    also have (\lambdam. \sumk=m*n..<m*n+n. \chik* of-nat k powr - s)=
                                    (\lambdam. of-nat n powr -s*(\sumk=1..n. \chi k*(of-nat m + of-nat k /
of-nat n) powr - s))
    proof (rule ext, goal-cases)
    case (1 m)
    have}(\sumk=m*n..<m*n+n. \chi k* of-nat k powr - s)
                        (\sumk=0..<n. \chi (k+m*n)* of-nat ( }m*n+k)\mathrm{ powr - s)
            by (intro sum.reindex-bij-witness[of - \lambdak. k+m*n \lambdak.k-m*n]) auto
    also have ... = (\sumk=0..<n. \chi k* of-nat (m*n+k) powr - s)
```

```
    by (simp add: periodic-mult)
    also have \(\ldots=\left(\sum k=0 . .<n\right.\). \(\chi k *\) (of-nat \(m+\) of-nat \(k /\) of-nat \(\left.n\right)\) 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 \(n\)
                    using \(n\) by (simp add: divide-simps del: div-mult-self1 div-mult-self2
div-mult-self3 div-mult-self4)
                            also have \(\ldots\) powr \(-s=(\) of-nat \(m+o f\)-nat \(k /\) of-nat \(n)\) powr \(-s *\) of-nat
\(n\) powr -s
            by (rule powr-times-real) auto
            finally show? case by simp
    qed
    also have \(\ldots=\) of-nat \(n\) powr \(-s *\left(\sum k=0 . .<n . \chi k *\right.\) (of-nat \(m+\) of-nat \(k\)
/ of-nat n) powr - s)
            by (subst sum-distrib-left) (simp-all add: mult-ac)
    also have \(\left(\sum k=0 . .<n . \chi k *(o f-n a t m+o f-n a t k / o f-n a t n) p o w r-s\right)=\)
                                    ( \(\sum k=1 . .<n . \chi k *(\) of-nat \(m+\) of-nat \(k /\) of-nat \(n)\) powr \(\left.-s\right)\)
    by (intro sum.mono-neutral-right) (auto simp: Suc-le-eq)
    also have \(\ldots=\left(\sum k=1\right.\)..n. \(\chi k *\) (of-nat \(m+\) of-nat \(k /\) of-nat \(\left.n\right)\) 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 \(-s *\left(\sum k=1 . . n . \chi k *(o f-n a t m+o f\right.\)-real
(of-nat \(k /\) of-nat n)) powr \(-s)\) ) sums
                                    (of-nat n powr \(-s *\left(\sum k=1 . . n . \chi k *\right.\) 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 \(\ldots=\) Dirichlet-L \(n \chi s\) using assms by (auto simp: Dirichlet-L-def)
    finally have Dirichlet-L \(n \chi s=\left(\sum n . \chi n *\right.\) of-nat \(n\) powr \(\left.-s\right)\)
    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 \((f d s \chi) s\)
    by (simp add: eval-fds-def sums-iff powr-minus field-simps fds-nth-fds')
qed
```

lemma fds-abs-converges-weak: Re $s>1 \Longrightarrow f d s$-abs-converges $(f d s \chi) s$
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 $\chi) \leq 1$
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 $\mathfrak{R}(s)>1$.

## lemma

fixes $s::$ complex assumes $s$ : Re $s>1$
shows Dirichlet-L-euler-product-LIMSEQ:
( $\lambda n$. . $p \leq n$. if prime $p$ then inverse $(1-\chi p /$ nat-power $p$ s) else 1) $\rightarrow$ Dirichlet-L $n \chi s$ (is ?th1)
and Dirichlet-L-abs-convergent-euler-product:
abs-convergent-prod ( $\lambda$ p. if prime $p$ then inverse $(1-\chi p / p$ powr $s)$
else 1)
(is ?th2)
proof -
have mult: completely-multiplicative-function (fds-nth (fds $\chi)$ )
using mult.completely-multiplicative-function-axioms by (simp add: fds-nth-fds')
have conv: fds-abs-converges ( $f d s \chi$ ) s
using abs-summable-Dirichlet-L[OF s]
by (simp add: fds-abs-converges-def fds-nth-fds' powr-minus divide-simps)
have $\left(\lambda n\right.$. $\prod p \leq n$. if prime $p$ then inverse $(1-\chi p /$ nat-power $p s)$ else 1) $\longrightarrow e v a l-f d s(f d s \chi) s$
using fds-euler-product-LIMSEQ' [OF mult conv] by (simp add: fds-nth-fds' cong: if-cong)
also have eval-fds (fds $\chi) s=$ 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.
from fds-abs-convergent-euler-product' [OF mult conv] show ?th2
by (simp add: fds-nth-fds cong: if-cong)
qed
lemma Dirichlet-L-Re-gt-1-nonzero:
assumes Re $s>1$
shows Dirichlet-L $n \chi s \neq 0$
proof -
have completely-multiplicative-function (fds-nth (fds $\chi)$ )
by (simp add: fds-nth-fds' mult.completely-multiplicative-function-axioms)
moreover have fds-abs-converges ( $f d s \chi$ ) s
using abs-summable-Dirichlet-L[OF assms]
by (simp add: fds-abs-converges-def fds-nth-fds' powr-minus divide-simps)
ultimately have (eval-fds $(f d s \chi) s=0) \longleftrightarrow(\exists$ p. prime $p \wedge f d s-n t h(f d s \chi) p$
$=$ nat-power $p s$ )
by (rule fds-abs-convergent-zero-iff)

```
    also have eval-fds \((f d s \chi) s=\) Dirichlet-L \(n \chi s\)
    using Dirichlet-L-conv-eval-fds-weak[OF assms] by simp
    also have \(\neg(\exists p\). prime \(p \wedge f d s\)-nth \((f d s \chi) p=\) nat-power \(p s)\)
    proof safe
    fix \(p::\) nat assume \(p\) : prime \(p\) fds-nth (fds \(\chi) p=\) 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 \(s\)
        by (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 \(p s=f d s-n t h(f d s \chi) p\) 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 \(x 0\) a \(b\) :: real and \(f f^{\prime}::\) real \(\Rightarrow\) real
    assumes nonprincipal: \(\chi \neq \chi_{0}\)
    assumes \(x 0: x 0 \geq 0\) and \(a b: x 0 \leq a a<b\)
    assumes \(f^{\prime}: \bigwedge x . x \geq x 0 \Longrightarrow\left(f\right.\) has-field-derivative \(\left.f^{\prime} x\right)(\) at \(x)\)
    assumes \(f\)-nonneg: \(\bigwedge x . x \geq x 0 \Longrightarrow f x \geq 0\)
    assumes \(f^{\prime}\)-nonpos: \(\bigwedge x . x \geq x 0 \Longrightarrow f^{\prime} x \leq 0\)
    shows \(\operatorname{norm}\left(\sum n \in\right.\) real \(-‘\{a<. . b\} . \chi n *(f(\) real \(\left.n))\right) \leq 2 *\) real (totient \(\left.n\right)\)
* \(f a\)
proof -
    note deriv \(=\) has-field-derivative-at-within \(\left[O F f^{\prime}\right]\)
    let ? \(A=\) sum-upto \(\chi\)
    have cont: continuous-on \(\{a . . b\} f\)
    by (rule DERIV-continuous-on[OF deriv]) (use ab in auto)
    have \(I^{\prime}:\left(f^{\prime}\right.\) has-integral \(\left.(f b-f a)\right)\{a . . b\}\)
    using \(a b\) deriv by (intro fundamental-theorem-of-calculus)
                (auto simp: has-real-derivative-iff-has-vector-derivative [symmetric])
    define \(I\) where \(I=\) integral \(\{a . . b\}\left(\lambda t\right.\). ? \(A t *\) of-real \(\left.\left(f^{\prime} t\right)\right)\)
    define \(C\) where \(C=\) real (totient \(n\) )
    have \(C\)-nonneg: \(C \geq 0\) by (simp add: \(C\)-def)
    have \(C\) : norm \((? A x) \leq C\) for \(x\)
    proof -
        have ? \(A x=\left(\sum k \leq n a t\lfloor x\rfloor \cdot \chi k\right)\) 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 \(\left.t * f^{\prime} t\right)\) has-integral ? A \(b * f b-\) ?A \(a * f a-\)
        \(\left(\sum n \in\right.\) real \(-‘\{a<. . b\} . \chi n * f(\) real \(\left.\left.n)\right)\right)\{a . . b\}\) using ab x0 cont \(f^{\prime}\)
```

```
    by (intro partial-summation-strong[of {}] has-vector-derivative-of-real) auto
    hence (\sumn\inreal -'{a<..b}.\chin*f(real n))=?A b*fb-?A a*fa-I
    by (simp add: has-integral-iff I-def)
    also have norm ... \leqnorm (?A b)* norm (fb) + norm (?A a)* norm (f a)
+ norm I
    by (rule order.trans[OF norm-triangle-ineq4] add-mono)+(simp-all add: norm-mult)
    also have norm I \leqintegral {a..b} ( }\lambdat\mathrm{ . of-real ( }-C)*\mathrm{ of-real ( }\mp@subsup{f}{}{\prime}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* (fb-fa))
    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*(fa-fb) by (simp add: algebra-simps)
    also have norm (sum-upto \chi b) \leqC by (rule C)
    also have norm (sum-upto \chi a) \leqC by (rule C)
    also have C*\operatorname{norm}(fb)+C*\operatorname{norm}(fa)+C*(fa-fb)=2*C*fa
    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 f
    assumes nonprincipal: }\chi\not=\mp@subsup{\chi}{0}{
    assumes }\mp@subsup{f}{}{\prime}:\x.x\geqx0\Longrightarrow(f has-field-derivative f'x)(at x
    assumes f-nonneg: \x. x\geqx0\Longrightarrowfx\geq0
    assumes f'-nonpos: \x. x \geqx0\Longrightarrow " f'x\leq0
    assumes lim:(f\longrightarrow0) at-top
    shows summable ( }\lambdan.\chin*fn
proof (rule summable-bounded-partials [where ?g = \lambdax. 2 * real (totient n)*f
x],goal-cases)
    case 1
    from eventually-ge-at-top[of nat \lceilx0\rceil] show ?case
    proof eventually-elim
    case (elim x)
    show ?case
    proof (safe, goal-cases)
        case (1 a b)
        with elim have *: max 0x0 \geq0 max 0x0 \leqa real a< real b
            by (simp-all add: nat-le-iff ceiling-le-iff)
        have }(\sumn\in{a<..b}.\chin* complex-of-real (f(real n)))
                    (\sumn\inreal -'{ real a<..real b}. \chi n* complex-of-real (f (real n)))
            by (intro sum.cong refl) auto
            also have norm \ldots\leq2 * real (totient n)*fa
            using nonprincipal * f'f
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 $s:$ real
assumes nonprincipal: $\chi \neq \chi_{0}$
shows conv-abscissa $(f d s \chi) \leq 0$
proof (rule conv-abscissa-leI)
fix $s$ assume $s: 0<$ ereal $s$
have summable ( $\lambda n . \chi n *$ of-real ( $n$ powr $-s$ ))
proof (rule summable-dcharacter-antimono[of 1])
fix $x::$ real assume $x \geq 1$
thus $((\lambda x . x$ powr $-s)$ has-field-derivative $(-s * x \operatorname{powr}(-s-1)))($ at $x)$
by (auto intro!: derivative-eq-intros)
qed (insert s assms, auto intro!: tendsto-neg-powr filterlim-ident)
thus $\exists s^{\prime}::$ complex. $s^{\prime} \cdot 1=s \wedge f d s$-converges $(f d s \chi) s^{\prime}$ 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')

qed
lemma summable-Dirichlet- $L^{\prime}$ :
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 $f d s$-converges ( $f d s \chi$ ) $s$
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: $\wedge s$. Re $s>0 \Longrightarrow$ Dirichlet-L $n \chi s=$ eval-fds
(fds $\chi$ ) s
and holomorphic-Dirichlet-L: Dirichlet-L $n$ र holomorphic-on A proof -
show eq: Dirichlet-L $n \chi s=$ eval-fds $(f d s \chi) s($ 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 $(\{s .0<R e s\}-\{1\})$
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 $n \chi$ holomorphic-on UNIV
proof (rule no-isolated-singularity')
from $n$ show Dirichlet-L $n \chi$ holomorphic-on (UNIV - $\{1\}$ )
by (intro holomorphic-Dirichlet-L-weak) auto
next
fix $s::$ complex assume $s: s \in\{1\}$
show Dirichlet-L $n \chi-s \rightarrow$ Dirichlet-L $n \chi s$
proof (rule Lim-transform-eventually)
from assms have continuous-on $\{s$. Re $s>0\}$ (eval-fds (fds $\chi)$ )
by (intro holomorphic-fds-eval holomorphic-on-imp-continuous-on) (auto intro: le-less-trans[OF conv-abscissa-le-0])
hence eval-fds (fds $\chi$ ) $-s \rightarrow$ eval-fds (fds $\chi$ ) susing $s$
by (subst (asm) continuous-on-eq-continuous-at) (auto simp: open-halfspace-Re-gt isCont-def)
also have eval-fds $(f d s \chi) s=$ Dirichlet-L $n \chi s$
using assms $s$ by (simp add: Dirichlet-L-def)
finally show eval-fds $(f d s \chi)-s \rightarrow$ Dirichlet-L $n \chi s$.
next
have eventually $(\lambda z . z \in\{z . \operatorname{Re} z>0\})(n h d s$ s) using $s$
by (intro eventually-nhds-in-open) (auto simp: open-halfspace-Re-gt)
hence eventually ( $\lambda z . z \in\{z . \operatorname{Re} z>0\}$ ) (at s)
unfolding eventually-at-filter by eventually-elim auto
then show eventually ( $\lambda$ z. eval-fds (fds $\chi) z=$ Dirichlet-L $n \chi z$ ) (at s)
by eventually-elim (auto intro!: eq [symmetric])
qed
qed auto
thus Dirichlet-L $n \chi$ holomorphic-on $A$ by (rule holomorphic-on-subset) auto qed
lemma cnj-Dirichlet-L:
cnj $($ Dirichlet-L $n \chi s)=$ Dirichlet-L $n($ inv-character $\chi)(c n j s)$
proof -
\{
assume $*: \chi \neq \chi_{0} s=1$
with summable-Dirichlet-L'[of 1] have ( $\lambda n$. $\chi$ n / $n$ ) sums eval-fds (fds $\chi$ ) 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 $\chi)) 1=\operatorname{cnj}(e v a l-f d s(f d s \chi) 1)$
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^1\not\inA
    shows Dirichlet-L n \chi holomorphic-on A
    using assms(2)
proof
    assume \chi = principal-dchar n ^1 &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>1f holomorphic-on A
        \chi}=\mathrm{ principal-dchar }n\wedge\mathrm{ dcharacter }n\chi\vee\chi=\mathrm{ principal-dchar }n\wedge(\forallx\inA
fx\not=1)
    shows (\lambdas. Dirichlet-L n \chi (f s)) holomorphic-on A
    using holomorphic-on-compose[OF assms(2) holomorphic-Dirichlet-L[OF assms(1),
of \chi]] assms
    by (auto simp: o-def image-iff)
lemma continuous-on-Dirichlet-L:
    assumes n>1 \chi\not= principal-dchar n ^ dcharacter n \chi\vee\chi= principal-dchar
n^1\not\inA
    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>1
    and \chi\not= principal-dchar n ^ dcharacter n \chi \vee \chi = principal-dchar n ^
( }\forallx\inA.fx\not=1
    shows continuous-on A ( }\lambdax\mathrm{ . Dirichlet-L n }\chi(fx)
    using continuous-on-compose\mathcal{Z}[OF continuous-on-Dirichlet-L[of n \chi f'A] assms(1)]
assms
    by (auto simp: image-iff)
corollary continuous-Dirichlet-L [continuous-intros]:
    n>1\Longrightarrow\chi\not= principal-dchar n}\wedge\mathrm{ dcharacter }n\chi\vee\chi=\mathrm{ principal-dchar n}\wedge
\not=1\Longrightarrow
        continuous (at s within A) (Dirichlet-L n \chi)
    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}
        \chi = principal-dchar n ^ dcharacter n \chi \vee \chi = principal-dchar n ^fs\not=1
\Longrightarrow
        continuous (at s within A) ( }\lambdax\mathrm{ . Dirichlet-L n 义 ( }fx)\mathrm{ )
    by (rule continuous-within-compose3[OF continuous-Dirichlet-L]) auto
```


## context residues-nat <br> begin

Applying the above to the $L\left(\chi_{0}, s\right)$, the $L$ function of the principal character, we find that it differs from the Riemann $\zeta$ function only by multiplication with a constant that depends only on the modulus $n$. They therefore have the same analytic properties as the $\zeta$ function itself.

```
lemma Dirichlet-L-principal:
    fixes \(s\) :: complex
    shows Dirichlet-L \(n \chi_{0} s=\left(\prod p \mid\right.\) prime \(p \wedge p\) dvd \(n .(1-1 / p\) powr \(\left.s)\right) *\)
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 \(\{s . \operatorname{Re} s>1\} \subseteq-\{1\}\) by auto
        show ?f \(s=\) ? \(g\) s if \(s \in\{s\). Re \(s>1\}\) for \(s\)
        proof -
            from that have \(s\) : Re \(s>1\) by simp
            let \(? P=\left(\prod p \mid\right.\) prime \(p \wedge p\) dvd \(n .(1-1 / p\) powr \(\left.s)\right)\)
            have \(\left(\lambda n\right.\). \(\eta \leq n\). if prime \(p\) then inverse \(\left(1-\chi_{0} p /\right.\) nat-power \(p\) s) else 1)
                                    \(\rightarrow\) Dirichlet-L \(n \chi_{0} s\)
            using \(s\) by (rule principal.Dirichlet-L-euler-product-LIMSEQ)
            also have ?this \(\longleftrightarrow\left(\lambda n\right.\). ? \(P *\left(\prod p \leq n\right.\). if prime \(p\) then inverse \((1-1 /\)
of-nat \(p\) powr s) else 1))
                                    \(\longrightarrow\) Dirichlet-L \(n \chi_{0} s\) (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 \(\left(\prod p \leq m\right.\). if prime \(p\) then inverse \(\left(1-\chi_{0} p /\right.\) nat-power \(p\) s) else 1\()=\)
                    \(\left(\prod p \mid p \leq m \wedge\right.\) prime \(p \wedge\) coprime \(p n\). ?f \(\left.p\right)(\) is \(-=\operatorname{prod}-\) ?A)
                    by (intro prod.mono-neutral-cong-right) (auto simp: principal-dchar-def)
            also have \(? A=\{p . p \leq m \wedge\) prime \(p\}-\{p\). prime \(p \wedge p\) dvd \(n\}\)
                    (is - = ? \(B-\) ? \(C\) ) using \(n\) by (auto dest: prime-imp-coprime simp:
coprime-absorb-left)
            also \{
            have \(*:\left(\prod p \in\right.\) ?B. ?f \(\left.p\right)=\left(\prod p \in\right.\) ?B - ?C. ?f \(\left.p\right) *\left(\prod p \in\right.\) ?C. ?f \(\left.p\right)\)
                    using \(1 n\) by (intro prod.subset-diff) (auto dest: dvd-imp-le)
                    have \(\left(\prod p \in\right.\) ?B. ?f \(\left.p\right) *\) ? \(P=\left(\prod p \in\right.\) ? \(B-\) ?C. ?f \(\left.p\right) *\left(\left(\prod p \in\right.\right.\) ? \(C\). ?f \(\left.p\right) *\)
?P)
            by (subst *) (simp add: mult-ac)
            also have \(\left(\prod p \in ? C . ? f p\right) * ? P=\left(\prod p \in ? C .1\right)\)
                    by (subst prod.distrib [symmetric], rule prod.cong)
                    (insert \(s\), auto simp: divide-simps powr-def exp-eq-1)
            also have \(\ldots=1\) by simp
            finally have \(\left(\prod p \in\right.\) ?B - ?C. ?f \(\left.p\right)=\left(\prod p \in\right.\) ?B. ?f \(\left.p\right) *\) ? \(P\) by \(\operatorname{sim} p\)
```

```
            }
            also have (\Pip\in?B. ?f p)=(\prodp\leqm. 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 \chi \chi s .
    moreover have ?g\longrightarrow ?P * zeta s
        by (intro tendsto-mult tendsto-const euler-product-zeta s)
        ultimately show Dirichlet-L n \chi}\mp@subsup{\chi}{0}{}s=?P*\mathrm{ zeta s
        by (rule LIMSEQ-unique)
    qed
qed (insert «s \not= 1` n, auto intro!: 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 $\mathfrak{R}(s) \geq 1$

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 $\neg p d v d n$ 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
qed
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 $\mathfrak{R}(s) \geq 1$.

```
theorem (in residues-nat) Dirichlet-L-Re-ge-1-nonzero-principal:
    assumes Re s \geq1s\not=1
    shows Dirichlet-L n (principal-dchar n) s\not=0
proof -
    have (\prodp| prime p ^ pdvd n. 1-1 / p powr s) \not=(0 :: complex )
    proof (subst prod-zero-iff)
        from n show finite {p. prime p}\wedgep\mathrm{ dvd n} by (intro finite-prime-divisors)
auto
        show }\neg(\existsp\in{p.prime p\wedgepdvdn}.1-1/p powr s=0
        proof safe
            fix p assume p: prime p pdvd n and 1-1/p powr s=0
            hence 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-gt-0-nat)
        qed
    qed
    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 nonreal 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 $\zeta$ 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\not=\mp@subsup{\chi}{0}{}\mathrm{ and Re u}\geq
    shows Dirichlet-L n \chiu\not=0
proof (cases Re u>1)
    include dcharacter-syntax
    case False
    define }a\mathrm{ where }a=-Im 
    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 }\mp@subsup{\chi}{}{\prime}\mathrm{ where [simp]: }\mp@subsup{\chi}{}{\prime}=\mathrm{ inv-character }
```

        - 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 \(\zeta\) function for \(\mathfrak{R}(s) \geq 1\), we define \(Q(s)=Z(s) Z(s+i a) Z(s-i a)\).
        Our objective is to show that the Dirichlet series of this function \(Q\) converges
        everywhere.
    define \(Z\) where \(Z=(\lambda s\). \(\Pi \chi \in\) dcharacters \(n\). Dirichlet-L \(n \chi s)\)
    define \(Z\) - \(f d s\) where \(Z\) - \(f d s=\left(\prod \chi \in\right.\) dcharacters \(n\). \(\left.f d s \chi\right)\)
    define \(Q\) where \(Q=\left(\lambda s . Z{ }_{s} \widehat{2} * Z(s+\mathrm{i} * a) * Z(s-\mathrm{i} * a)\right)\)
    define \(Q\)-fds where \(Q\)-fds \(=Z\) - ffs \({ }^{\wedge} 2 * f d s\)-shift \((\mathrm{i} * a) Z\)-fds \(*\) fds-shift \((-\mathrm{i}\)
    * a) Z-fds
let ? sings $=\{1,1+\mathrm{i} * a, 1-\mathrm{i} * a\}$
    - Some preliminary auxiliary facts
define $P$ where $P=\left(\lambda s\right.$. $\left(\prod x \in\{p\right.$. prime $p \wedge p d v d n\} .1-1 /$ of-nat $x$ powr
$s$ :: complex))
have $\chi_{0}: \chi_{0} \in$ dcharacters $n$ by (auto simp: principal.dcharacter-axioms dchar-
acters-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 $n \chi^{\prime} \in$ dcharacters $n$ using dcharacter-axioms
by (auto simp add: dcharacters-def dcharacter.dcharacter-inv-character)
from zero dcharacter.cnj-Dirichlet-L[of $n \chi 1-\mathrm{i} * a]$ dcharacter-axioms
have zero': Dirichlet-L $n \chi^{\prime}(1+\mathrm{i} * a)=0$ by $\operatorname{simp}$
have $Z=\left(\lambda s\right.$. Dirichlet-L $n \chi_{0} s *\left(\prod \chi \in\right.$ dcharacters $n-\left\{\chi_{0}\right\}$. Dirichlet- $L n$ $\chi s)$ )
unfolding $Z$-def using $\chi_{0}$ by (intro ext prod.remove) auto
also have $\ldots=\left(\lambda s . P s *\right.$ zeta $s *\left(\prod \chi \in\right.$ dcharacters $n-\left\{\chi_{0}\right\}$. Dirichlet- $L n$ $\chi s)$ )
by (simp add: Dirichlet-L-principal P-def)
finally have $Z$-eq: $Z=\left(\lambda s . P s *\right.$ zeta $s *\left(\Pi \chi \in\right.$ dcharacters $n-\left\{\chi_{0}\right\}$. Dirichlet-L $n \chi s)$ ).
have $Z$-eq': $Z=(\lambda s . P s *$ zeta $s *$ Dirichlet-L $n \chi s *$
( $\prod_{\chi} \in$ dcharacters $n-\left\{\chi_{0}\right\}-\{\chi\}$. Dirichlet-L $\left.n \chi s\right)$ )
if $\chi \in$ dcharacters $n \chi \neq \chi_{0}$ for $\chi$
proof (rule ext, goal-cases)
case (1 s)
from that have $\chi: \chi \in$ dcharacters $n$ by (simp add: dcharacters-def)
have $Z s=P s *$ zeta $s *$
( $\Pi \chi \in$ dcharacters $n-\left\{\chi_{0}\right\}$. Dirichlet- $L n \chi$ s) by (simp add: Z-eq)
also have $\left(\Pi \chi \in\right.$ dcharacters $n-\left\{\chi_{0}\right\}$. Dirichlet- $L n \chi$ s) $=$ Dirichlet- $L n \chi$ $s *$

$$
\left(\prod \chi \in \text { dcharacters } n-\left\{\chi_{0}\right\}-\{\chi\} . \text { Dirichlet-L } n \chi \quad s\right)
$$

using assms $\chi$ that by (intro prod.remove) auto
finally show ?case by (simp add: mult-ac)
qed

- 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: $Q \in O[a t s](\lambda$-. 1$)$ for $s$
proof (cases $a=0$ )
case True
have $(\lambda s$. Dirichlet-L $n \chi s-$ Dirichlet-L $n \chi 1) \in O[$ at 1$](\lambda s . s-1)$ using $\chi$ 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 $n \chi s *$
$\left(\prod \chi \in d\right.$ characters $n-\left\{\chi_{0}\right\}-\{\chi\}$. Dirichlet-L $\left.n \chi s\right)$ )
using $\chi$ 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 continu-ous-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 $Z \in O[a t s](\lambda-.1)$
proof (cases $s=1$ )
case False
thus ?thesis unfolding Z-def using $n \chi$
by (intro continuous-imp-bigo-1 continuous-intros) (auto simp: dcharac-ters-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 $\ldots \in O[a t s](\lambda-.1 * 1 * 1 * 1)$
by (intro landau-o.big.mult) fact+
finally show?thesis by simp
next
case False
have bigo1: $(\lambda s . Z s * Z(s-\mathrm{i} * a)) \in O[$ at 1$](\lambda-.1)$
if Dirichlet- $L n \chi(1-\mathrm{i} * a)=0 a \neq 0 \chi \in$ dcharacters $n \chi \neq \chi_{0}$
for $a$ :: real and $\chi$
proof -
have $(\lambda s$. Dirichlet-L $n \chi(s-\mathrm{i} * a)-$ Dirichlet-L $n \chi(1-\mathrm{i} * a)) \in O[a t$ $1](\lambda s . s-1)$
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 $n \chi(s-\mathrm{i} * a)) \in O[$ at 1$](\lambda s . s-1)$ using that by $\operatorname{simp}$
have $(\lambda s . Z(s-\mathrm{i} * a))=(\lambda s . P(s-\mathrm{i} * a) *$ zeta $(s-\mathrm{i} * a) *$ Dirichlet- $L n$ $\chi(s-\mathrm{i} * a)$ * $\left(\prod \chi \in\right.$ dcharacters $n-\left\{\chi_{0}\right\}-\{\chi\}$. Dirichlet-L $\left.\left.n \chi(s-\mathrm{i} * a)\right)\right)$
using that by (subst Z-eq'[of $\chi]$ ) auto
also have $\ldots \in O[a t 1](\lambda s .1 * 1 *(s-1) * 1)$ unfolding $P$-def using that $n$
by (intro landau-o.big.mult continuous-imp-bigo-1 continuous-intros *)
(auto simp: prime-gt-0-nat dcharacters-def)
finally have $(\lambda s . Z(s-\mathrm{i} * a)) \in O[a t 1](\lambda s . s-1)$ by $\operatorname{simp}$
moreover have $Z \in O[$ at 1$](\lambda s .1 *(1 /(s-1)) * 1)$ unfolding $Z-e q$ using $n$ that
by (intro landau-o.big.mult zeta-bigo-at-1 continuous-imp-bigo-1 continu-ous-intros)
(auto simp: dcharacters-def)
hence $Z \in O[$ at 1$](\lambda s .1 /(s-1))$ by simp
ultimately have $(\lambda s . Z s * Z(s-\mathrm{i} * a)) \in O[$ at 1$](\lambda s .1 /(s-1) *(s$ $-1)$ )
by (intro landau-o.big.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': $(\lambda s . Z s * Z(s+\mathrm{i} * a)) \in O[$ at 1$](\lambda-.1)$
if Dirichlet-L $n \chi(1-\mathrm{i} * a)=0 a \neq 0 \chi \in d$ characters $n \chi \neq \chi_{0}$
for $a$ :: real and $\chi$
proof -
from that interpret dcharacter $n G \chi$ by (simp-all add: dcharacters-def
G-def)
from bigo1[of inv-character $\chi-a]$ that cnj-Dirichlet- $L[$ of $1-\mathrm{i} * a$ ] show
?thesis
by (simp add: dcharacters-def dcharacter-inv-character)
qed
have bigo2: $(\lambda s . Z s * Z(s-\mathrm{i} * a)) \in O[a t(1+\mathrm{i} * a)](\lambda-.1)$
if Dirichlet-L $n \chi(1-\mathrm{i} * a)=0 a \neq 0 \chi \in$ dcharacters $n \chi \neq \chi_{0}$
for $a::$ real and $\chi$
proof -
have $(\lambda s . Z s * Z(s-\mathrm{i} * a)) \in O[$ filtermap $(\lambda s . s+\mathrm{i} * a)($ at 1$)](\lambda-.1)$
using bigo1' $[$ of $\chi$ a] that by (simp add: mult.commute landau-o.big.in-filtermap-iff)
also have filtermap $(\lambda s . s+\mathrm{i} * a)($ at 1$)=a t(1+\mathrm{i} * a)$
using filtermap-at-shift[of -i * a 1] by simp
finally show ?thesis .
qed
have bigo2': $(\lambda s . Z s * Z(s+\mathrm{i} * a)) \in O[$ at $(1-\mathrm{i} * a)](\lambda-.1)$
if Dirichlet-L $n \chi(1-\mathrm{i} * a)=0 a \neq 0 \chi \in$ dcharacters $n \chi \neq \chi_{0}$
for $a$ :: real and $\chi$
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 $1-\mathrm{i} * a]$ show
?thesis
by (simp add: dcharacters-def dcharacter-inv-character)
qed
have $Q$-eq: $Q=(\lambda s .(Z s * Z(s+\mathrm{i} * a)) *(Z s * Z(s-\mathrm{i} * a)))$
by (simp add: Q-def power2-eq-square mult-ac)

```
consider \(s=1|s=1+\mathrm{i} * a| s=1-\mathrm{i} * a \mid s \notin\) ? sings by blast
thus ?thesis
proof cases
    case 1
    have \(Q \in O[\) at 1\(](\lambda-.1 * 1)\)
            unfolding \(Q\)-eq using assms zero zero' False \(\chi\)
            by (intro landau-o.big.mult bigo1[of \(\chi\) a] bigo1' \([\) of \(\chi\) a \(]\); simp \()+\)
        with 1 show ?thesis by simp
    next
        case 2
```

```
            have Q GO[at (1+i *a)](\lambda-. 1*1) unfolding Q-eq
                    using assms zero zero' False \chi n
                    by (intro landau-o.big.mult bigo2[of \chi a] continuous-imp-bigo-1)
                    (auto simp: Z-def dcharacters-def intro!: continuous-intros)
            with 2 show ?thesis by simp
    next
            case 3
            have Q}\inO[at(1-\textrm{i}*a)](\lambda-.1*1)\mathrm{ unfolding }Q\mathrm{ -eq
                using assms zero zero' False \chi n
                by (intro landau-o.big.mult bigo2'[of \chi a] continuous-imp-bigo-1)
                    (auto simp: Z-def dcharacters-def intro!: continuous-intros)
            with 3 show ?thesis by simp
        next
            case 4
            thus ?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 $\exists Q^{\prime} . Q^{\prime}$ holomorphic-on UNIV $\wedge\left(\forall z \in U N I V-\right.$ ? sings. $\left.Q^{\prime} z=Q z\right)$
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 $Q^{\prime}$ where $Q^{\prime}: Q^{\prime}$ holomorphic-on UNIV $\wedge z . z \notin$ ?sings $\Longrightarrow Q^{\prime}$
$z=Q z$ by blast
    - $Q^{\prime}$ constitutes an analytic continuation of the Dirichlet series of $Q$.
have eval- $Q$-fds: eval-fds $Q$-fds $s=Q^{\prime} s$ if Re $s>1$ for $s$
proof -
have [simp]: dcharacter $n \chi$ if $\chi \in$ dcharacters $n$ for $\chi$
using that by (simp add: dcharacters-def)
from that have abs-conv-abscissa (fds $\chi)<\operatorname{ereal}$ (Re s) if $\chi \in$ dcharacters
$n$ for $\chi$
using that by (intro le-less-trans[OF dcharacter.abs-conv-abscissa-weak[of
$n \chi]]$ ) 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 $\ldots=Q^{\prime}$ s by (subst $Q^{\prime}$ ) 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 $[k=1](\bmod n)$ then totient $n$ else $0::$ real $)$
have $\ln -Q$-fds-eq:
$f d s-\ln 0 Q$-fds $=f d s(\lambda k$. of-real $(2 * I k *$ mangoldt $k / \ln k *(1+\cos (a *$ $\ln k))$ )


## proof -

have $n z: \chi($ Suc 0$)=1$ if $\chi \in d$ characters $n$ for $\chi$
using dcharacter.Suc- $0[$ of $n \chi]$ that by (simp add: dcharacters-def)
note simps $=f d s$ - $l n$-mult $\left[\right.$ where $l^{\prime}=0$ and $\left.l^{\prime \prime}=0\right] f d s$-ln-power $\left[\right.$ where $l^{\prime}$ $=0]$

$$
f d s-l n-p r o d\left[\text { where } l^{\prime}=\lambda-.0\right]
$$

have $f d s-\ln 0 Q-f d s=\left(\sum \chi \in d c h a r a c t e r s ~ n . ~ 2 * ~ f d s-l n ~ 0(f d s ~ \chi)+\right.$
$f d s$-shift $(\mathrm{i} * a)(f d s-\ln 0(f d s \quad \chi))+f d s-s h i f t(-\mathrm{i} * a)(f d s-\ln 0(f d s \chi)))$
by (auto simp: $Q$-fds-def $Z$-fds-def simps nz sum.distrib sum-distrib-left)
also have $\ldots=\left(\sum \chi \in\right.$ dcharacters $n$.
fds $(\lambda k . \chi k *$ of-real (2 $*$ mangoldt $k / \ln k *(1+\cos (a *$ ln $k)$ ))))
(is $\left(\sum \chi \in-\right.$. ?l $\left.\chi\right)=-$ )
proof (intro sum.cong refl, goal-cases)
case (1 $\chi$ )
then interpret dcharacter $n G \chi$ by (simp-all add: dcharacters-def $G$-def)
have mult: completely-multiplicative-function (fds-nth (fds $\chi)$ )
by (simp add: fds-nth-fds' mult.completely-multiplicative-function-axioms)
have $*: f d s-\ln 0(f d s \chi)=f d s(\lambda n . \chi n * m a n g o l d t n / R \ln ($ real $n))$
by (simp add: fds-ln-completely-multiplicative[OF mult] fds-nth-fds'
fds-eq-iff)
have ?l $\chi=f d s(\lambda k . \chi k *$ mangoldt $k / R \ln k *(2+k$ powr $(\mathrm{i} * a)+k$ powr $(-\mathrm{i} * a))$ )
by (unfold *, rule fds-eqI) (simp add: algebra-simps scaleR-conv-of-real numeral-fds)
also have $\ldots=f d s(\lambda k . \chi k * 2 *$ mangoldt $k / R \ln k *(1+\cos (o f-r e a l(a$ * $\ln k)$ )) )
unfolding cos-exp-eq by (intro fds-eqI) (simp add: powr-def algebra-simps)
also have $\ldots=f d s(\lambda k . \chi k *$ 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 $\ldots=f d s\left(\lambda k .\left(\sum \chi \in d\right.\right.$ characters $\left.n . \chi k\right) *$ of-real $(2 *$ mangoldt $k / \ln k *$

$$
(1+\cos (a * \ln k))))
$$

by (simp add: sum-distrib-right sum-divide-distrib scale $R$-conv-of-real sum-distrib-left)
also have $\ldots=f d s(\lambda k$. of-real $(2 * I k *$ mangoldt $k / \ln k *(1+\cos (a$ * $\ln k)$ ))
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 nonneg-dirichlet-series ( $f d s-l n 0 \quad Q$ - $f d s$ )
proof

```
    show fds-nth (fds-ln 0 Q-fds)k\in\mp@subsup{\mathbb{R}}{\geq0}{}\mathrm{ for }k
    proof (cases k<2)
    case False
    have cos: 1+ cos x\geq0 for x :: real
        using cos-ge-minus-one[of x] by linarith
    have fds-nth (fds-ln 0 Q-fds)k=
                of-real (2*Ik* mangoldt k/lnk*(1+\operatorname{cos}(a*\operatorname{ln}k)))
        by (auto simp: fds-nth-fds' ln-Q-fds-eq)
    also have ... \in\mathbb{R}\geq0}\mathrm{ 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 nonneg: nonneg-dirichlet-series Q-fds
proof (rule nonneg-dirichlet-series-lnD)
    have (\prodx\indcharacters 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 $\leq 1$ 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 〈 $Q^{\prime}$ holomorphic-on UNIV〉
have abscissa: abs-conv-abscissa $Q$-fds $=-\infty$
by (intro entire-continuation-imp-abs-conv-abscissa-MInfty[where $g=Q^{\prime}$
and $c=1]$ )
(auto simp: one-ereal-def)
- Again, similarly to the proof for $\zeta$, 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 $=f d s$-primepow-subseries $p Q$ - $f d s$
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 $\leq a b s$-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' ${ }^{\prime}$ 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 $Z(s+i a)$. It is clear that it is entire and non-zero for
$\mathfrak{R}(s)>0$ and all real $a$.
define $g$ :: real $\Rightarrow$ complex $\Rightarrow$ complex
where $g=(\lambda a s$. ( $\rceil \chi \in$ dcharacters $n .(1-\chi p * p$ powr $(-s+\mathrm{i} *$ of-real
a) )) )
have $g$ - $n z: g$ a $s \neq 0$ if Res>0 for sa unfolding $g$-def
proof (subst prod-zero-iff[OF finite-dcharacters], safe)
fix $\chi$ assume $\chi \in$ dcharacters $n$ and $*: 1-\chi p * p$ powr $(-s+\mathrm{i} * a)=0$
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 $0<\operatorname{norm}(1::$ complex $)-\operatorname{norm}(\chi p * p \operatorname{powr}(-s+\mathrm{i} * a))$
using $p$ by (simp add: norm-mult norm norm-powr-real-powr)
also have $\ldots \leq \operatorname{norm}(1-\chi p * p \operatorname{powr}(-s+\mathrm{i} * a)$ )
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

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

for every $s$ with $\mathfrak{R}(s)>1$, and by analytic continuation also for $\mathfrak{R}(s)>0$.
have eval-R: eval-fds $R$-fds $s=1 /(g 0 s$ へ $2 * g a s * g(-a) s)$
(is $-=$ ?f $s$ ) if Re $s>0$ for $s::$ complex
proof -
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
$-a]$
by (intro holomorphic-intros) (auto simp: g-nz)
next
fix $z$ assume $z: z \in\{s$. Re $s>1\}$
have [simp]: completely-multiplicative-function $\chi$ fds-nth $(f d s \chi)=\chi$
if $\chi \in$ dcharacters $n$ for $\chi$ proof -
from that interpret dcharacter $n G \chi$ by (simp-all add: G-def dcharac-ters-def)
show completely-multiplicative-function $\chi$ fds-nth $(f d s \chi)=\chi$ by (simp-all add: fds-nth-fds' mult.completely-multiplicative-function-axioms) qed have $[$ simp $]$ : dcharacter $n \chi$ if $\chi \in$ dcharacters $n$ for $\chi$
using that by (simp add: dcharacters-def)
from that have abs-conv-abscissa $(f d s \chi)<\operatorname{ereal}(\operatorname{Re} z)$ if $\chi \in$ dcharacters
$n$ for $\chi$ 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 $z p$
by (simp add: R-fds-def $Q$-fds-def Z-fds-def eval-fds-mult eval-fds-prod eval-fds-power
fds-abs-converges-mult fds-abs-converges-power fds-abs-converges-prod $g$-def mult-ac
fds-primepow-subseries-euler-product-cm powr-minus powr-diff powr-add prod-dividef
fds-abs-summable-zeta g-nz fds-abs-converges power-one-over di-vide-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 $\{s . \operatorname{Re} s>0\})$ 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 continu-ous-within-subset) auto
thus ?thesis by (auto simp: continuous-within)
qed
have $\left(\left(\lambda s . g 0 s{ }^{\wedge} 2 * g a s * g(-a) s\right) \longrightarrow g 00{ }^{\text {へ }} 2 * g a 0 * g(-a) 0\right)$ (at 0 within $\{s . \operatorname{Re} s>0\}$ ) by (intro tendsto-intros $g$-limit)
also have $g 00=0$ unfolding $g$-def
proof (rule prod-zero)
from $p$ and $\chi_{0}$ show $\exists \chi \in$ dcharacters $n .1-\chi p *$ of-nat $p$ powr $(-0+$ $\mathrm{i} *$ of-real 0$)=0$
by (intro bexI[of - $\left.\chi_{0}\right]$ ) (auto simp: principal-dchar-def)
qed auto
moreover have eventually $(\lambda s . s \in\{s . \operatorname{Re} s>0\})$ (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 $(\lambda s . g 0 s \wedge 2 * g a s * g(-a) s)(a t 0)$
(at 0 within $\{s . R e s>0\}$ ) 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 ( $\lambda s . \operatorname{Re} s>0$ ) (at 0 within $\{s . \operatorname{Re} s>0\})$
by (auto simp: eventually-at intro!: exI[of-1])
have ?lim $\longleftrightarrow$ 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 ... .
    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 \(\{s . \operatorname{Re} s>0\}\) )
        by (auto simp: continuous-within)
    next
    have \(0 \in\{s\). Re \(s \geq 0\}\) by simp
    also have \(\{s\). Re \(s \geq 0\}=\) closure \(\{s\). Re \(s>0\}\)
        using closure-halfspace-gt[of 1::complex 0] by (simp add: inner-commute)
    finally have \(0 \in \ldots\).
    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-upto \(\chi x\) ) \(\leq\) totient \(n\)
proof -
    have sum-upto \(\chi x=\left(\sum k \leq n a t\lfloor x\rfloor \cdot \chi k\right)\) unfolding sum-upto-altdef
        by (intro sum.mono-neutral-left) auto
    also have norm \(\ldots \leq\) totient \(n\)
        by (rule sum-dcharacter-atMost-le) fact
    finally show ?thesis.
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 \(\sigma \equiv \operatorname{Re} s\)
    shows norm (sum-upto \((\lambda n . \chi n * n\) powr \(-s) x-\) Dirichlet-L \(n \chi s) \leq\)
                        real \((\) totient \(n) *(2+c \bmod s / \sigma) / x\) 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-nat \(n\) 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-nat \(k\) powr \(\left.-s\right) \longrightarrow\) Dirichlet-L \(n \chi s\)
```

```
    by (simp add: sums-def' atLeast0AtMost)
    thus (\lambdam. sum-upto ( }\lambdak.\chik*\mathrm{ of-nat k powr -s) x - ( }\sumk\leqm.\chik*of-nat 
powr -s))
            <sum-upto ( }\lambdak.\chik*\mathrm{ of-nat k powr -s) x - Dirichlet-L n र s
    by (intro tendsto-intros)
next
    define }M\mathrm{ where }M=\mathrm{ sum-upto }
    have le: norm ( }\sumn\in\mathrm{ real- `{x<..y}. र n* of-nat n powr - s)
                \leqreal (totient n)*(2+cmod s/\sigma)/ x powr \sigma if xy: 0< < < < < y
for x y
    proof -
    from xy have I: ((\lambdat.Mt*(-s*t powr (-s-1))) has-integral
                                    My*of-real y powr - s-M x* of-real x powr - s-
                                    (\sumn\inreal-`{x<..y}. \chi n* of-real (real n) powr -s)) {x..y}
unfolding M-def
            by (intro partial-summation-strong [of {}])
                (auto intro!: has-vector-derivative-real-field derivative-eq-intros continu-
ous-intros)
    hence (\sumn\inreal- `{x<..y}. \chi n* real n powr -s)=
                    My* of-real y powr - s-Mx* of-real x powr -s -
                    integral {x..y} (\lambdat.Mt* (-s*t powr (-s-1)))
        by (simp add: has-integral-iff)
    also have norm ... \leqnorm (My* of-real y powr -s)+norm (Mx* of-real
x powr -s) +
                        norm (integral {x..y} (\lambdat.Mt* (-s*t powr (-s-1))))
        by (intro order.trans[OF norm-triangle-ineq4] add-mono order.refl)
    also have norm (My* of-real y powr -s)\leqtotient n* y powr -\sigma
        using xy assms unfolding norm-mult M-def \sigma-def
    by (intro mult-mono sum-upto-dcharacter-le) (auto simp: norm-powr-real-powr)
    also have ... \leqtotient n*x powr -\sigma
        using assms xy by (intro mult-left-mono powr-mono2') (auto simp: \sigma-def)
    also have norm (Mx* of-real x powr -s)\leqtotient n*x powr -\sigma
        using xy assms unfolding norm-mult M-def \sigma-def
    by (intro mult-mono sum-upto-dcharacter-le) (auto simp: norm-powr-real-powr)
    also have norm (integral {x..y} (\lambdat.Mt* (-s* of-real t powr (-s-1))))\leq
                integral {x..y} (\lambdat. real (totient n)* norm s*t powr ( }-\sigma-1)
    proof (rule integral-norm-bound-integral integrable-on-cmult-left)
        show ( }\lambdat.\mathrm{ real (totient n)* norm s*t powr (- 
            using xy by (intro integrable-continuous-real continuous-intros) auto
    next
        fix }t\mathrm{ assume }t:t\in{x..y
        have norm (Mt*(-s* of-real t powr (-s-1))) \leq
                real (totient n) * (norm s*t powr (-\sigma-1))
            unfolding norm-mult M-def \sigma-def using xy t assms
    by (intro mult-mono sum-upto-dcharacter-le) (auto simp: norm-mult norm-powr-real-powr)
    thus norm (Mt* (-s* of-real t powr (-s-1)))\leqreal (totient n)* 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} (\lambdat.t powr (-\sigma-1))
        by simp
    also have ((\lambdat.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] \sigma-def)
    hence integral {x..y} (\lambdat.t powr (-\sigma-1)) = y powr -\sigma / (-\sigma) - x powr -\sigma
/ (-\sigma)
            by (simp add: has-integral-iff)
    also from assms have ... \leqx powr -\sigma/\sigma by (simp add: \sigma-def)
    also have real (totient n)*x powr -\sigma + real (totient n)* x powr }-\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 (\lambdam. norm (sum-upto ( }\lambdak.\chik*\mathrm{ of-nat k powr - s) x-
                    (\sumk\leqm. \chi k * of-nat k powr - s))
            \leqreal (totient n)*(2 + cmod s / \sigma) / x powr \sigma) at-top
    using eventually-gt-at-top[of nat \lfloorx\rfloor]
proof eventually-elim
    case (elim m)
    have ( }\sumk\leqm.\chik*of-nat k powr - s) - sum-upto ( \lambdak. \chi k* of-nat k powr
- s) x=
            ( \sumk\in{..m} - {k.0<k\wedge real k\leqx}. \chi k* of-nat k powr -s) unfolding
sum-upto-def
            using elim}\langlex>0\rangle\mathrm{ by (intro Groups-Big.sum-diff [symmetric])
                                    (auto simp: nat-less-iff floor-less-iff)
                            also have ... = (\sumk\in{..m} - {k. real k\leqx}. \chik* of-nat k powr -s)
            by (intro sum.mono-neutral-right) auto
    also have {..m} - {k. real k\leqx} = real -' {x<..real m}
            using elim \langlex > 0\rangle by (auto simp: nat-less-iff floor-less-iff not-less)
    also have norm ( }\sumk\in\ldots.\chi<k*\mathrm{ of-nat k powr -s)}
                        real (totient n)* (2 + cmod s / \sigma) / x powr \sigma
            using elim \langlex > 0\rangle by (intro le) (auto simp: nat-less-iff floor-less-iff)
    finally show ?case by (simp add: norm-minus-commute)
    qed
qed auto
lemma partial-Dirichlet-L-sum-bigo:
    fixes }s:: complex and x :: real
    assumes }\chi\not=\mp@subsup{\chi}{0}{}\mathrm{ Re s>0
    shows (\lambdax.sum-upto (\lambdan.\chin*n powr -s)x-Dirichlet-L n \chi s) \in O(\lambdax.
x powr -s)
proof (rule bigoI)
    show eventually ( }\lambdax.norm (sum-upto (\lambdan. \chi n* of-nat n powr -s) x - Dirich-
```

```
let-L n \chi s)
                \leqreal (totient n)*(2 + norm s/Re s)* norm (of-real x powr - s))
at-top
    using eventually-gt-at-top[of 0]
    proof eventually-elim
    case (elim x)
    have norm (sum-upto (\lambdan. \chi n* of-nat n powr -s) x - Dirichlet-L n \chi s)
                        \leqreal (totient n)*(2 + norm s / Re s)/x powr Re s
        using elim assms by (intro Dirichlet-L-minus-partial-sum-bound) auto
    thus ?case using elim assms
        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)
    qed
qed
end
```


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

context residues-nat
begin
lemma Dirichlet-L-0-principal [simp]: Dirichlet-L n $\chi_{0} 0=0$
proof -
have Dirichlet-L $n \chi_{0} 0=-1 / 2 *\left(\prod p \mid\right.$ prime $p \wedge p d v d n .1-1 / p$ powr 0)
by (simp add: Dirichlet-L-principal prime-gt-0-nat)
also have $\left(\prod p \mid\right.$ prime $p \wedge p$ dvd $n .1-1 / p$ powr 0$)=\left(\prod p \mid\right.$ prime $p \wedge p$ dvd n. 0 :: complex)
by (intro prod.cong) (auto simp: prime-gt-0-nat)
also have $\left(\prod p \mid\right.$ prime $p \wedge p d v d n .0::$ 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 $n \chi 0=-\left(\sum k=1 . .<n\right.$. of-nat $\left.k * \chi k\right) /$ of-nat $n$
proof -
have Dirichlet-L $n \chi 0=\left(\sum k=1 . . n . \chi k *(1 / 2-\right.$ of-nat $k /$ of-nat $\left.n)\right)$
using assms $n$ by (simp add: Dirichlet-L-conv-hurwitz-zeta-nonprincipal)
also have $\ldots=-1 / n *\left(\sum k=1\right.$..n. of-nat $\left.k * \chi k\right)$
using assms by (simp add: algebra-simps sum-subtractf sum-dcharacter-block' sum-divide-distrib [symmetric])
also have $\left(\sum k=1\right.$..n. of-nat $\left.k * \chi k\right)=\left(\sum k=1 . .<n\right.$. of-nat $\left.k * \chi k\right)$
using $n$ by (intro sum.mono-neutral-right) (auto simp: eq-zero-iff)
finally show eq: Dirichlet-L $n \chi 0=-\left(\sum k=1 . .<n\right.$. of-nat $\left.k * \chi k\right) /$ of-nat $n$ by $\operatorname{simp}$

```
qed
lemma Dirichlet-L-0-even [simp]:
    assumes }\chi(n-1)=
    shows Dirichlet-L n\chi0=0
proof (cases \chi = \chi )
    case False
    hence Dirichlet-L n \chi 0 = - (\sumk=Suc 0..<n. of-nat k*\chik)/ 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 - (\sumk=1..<n. of-nat k*\chi
k) / of-nat n)
    by (cases \chi = \chio) (auto simp: Dirichlet-L-0-nonprincipal)
end
```


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

```
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 (\chi k)) = \chi k
    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 (\chik)=\chik
    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 \chis\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>0
    shows sum-char-divisors-ge: Re (\sumd|d dvd k.\chid)\geq0(is Re(?A k)\geq0)
    and sum-char-divisors-square-ge: is-square k\LongrightarrowRe(\sumd|d dvd k. \chi d)\geq1
proof -
    interpret sum: multiplicative-function ?A
            by (fact mult.multiplicative-sum-divisors)
    have A:?A k\in\mathbb{R}\mathrm{ for }k\mathrm{ by (intro sum-in-Reals real)}
    hence [simp]: Im (?A k) = 0 for k by (auto elim!: Reals-cases)
    have *: Re (?A ( p^ m)) \geq(if even m then 1 else 0) if p: prime p for p m
    proof -
            have sum-neg1: (\sumi\leqm. (-1) ^i)=(if even m then 1 else (0::real))
            by (induction m) auto
            from p have ?A ( }\mp@subsup{p}{}{`}m)=(\sumk\leqm.\chi(\mp@subsup{p}{}{`}k)
            by (intro sum.reindex-bij-betw [symmetric] bij-betw-prime-power-divisors)
    also have Re... = (\sumk\leqm.Re (\chi p)^ ^) by (simp add:mult.power)
    also have ... \geq(if even m then 1 else 0)
            using sum-neg1 char-cases[of p] by (auto simp: power-0-left)
            finally show ?thesis .
    qed
```



```
p m
    using *[of p m] that by (auto split: if-splits)
    have eq: Re (?A k) = (\prodp\inprime-factors k. Re (?A ( p^ multiplicity p k)))
        using kA by (subst sum.prod-prime-factors) (auto simp: Re-prod-Reals)
    show Re (\sumd |d dvd k. \chid)\geq0 by (subst eq, intro prod-nonneg ballI *) auto
    assume is-square k
    then obtain m where m:k=m^2 by (auto elim!: is-nth-powerE)
    have even (multiplicity pk) if prime p for p using k that unfolding m
        by (subst prime-elem-multiplicity-power-distrib) (auto intro!: Nat.grOI)
    thus Re (\sumd|d dvd k.\chid)\geq1
        by (subst eq, intro prod-ge-1 ballI *) auto
qed
end
```


## 4 Dirichlet's Theorem on primes in arithmetic progressions

theory Dirichlet-Theorem imports<br>Dirichlet-L-Functions<br>Bertrands-Postulate.Bertrand<br>Landau-Symbols.Landau-More<br>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 \(x \geq 0\)
    shows sum-upto mangoldt \(x \leq 3 / 2 * x\)
proof -
    have sum-upto mangoldt \(x=\) psi (nat \(\lfloor x\rfloor)\)
        by (simp add: sum-upto-altdef psi-def atLeastSucAtMost-greaterThanAtMost)
    also have \(\ldots \leq 551 / 256 * \ln 2 *\) real \((\) nat \(\lfloor x\rfloor)\)
        by (rule psi-ubound-log)
    also have \(\ldots \leq 3 / 2 *\) real (nat \(\lfloor x\rfloor)\)
        using Bertrand.ln-2-le by (intro mult-right-mono) auto
    also have \(\ldots \leq 3 / 2 * x\) using assms by linarith
    finally show ?thesis.
qed
```

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:
    (\lambdax. (\sump| prime p ^ real p\leqx. ln p/p) - ln x) \inO(\lambda-. 1)
proof (intro bigoI[of - 8] eventually-mono[OF eventually-gt-at-top[of 1]], goal-cases)
    case (1 x)
    have |(\sump| prime p}\wedgep\leqnat \lfloorx\rfloor. ln p/p)-\operatorname{ln}x|
    |(\sump|prime p}\wedge p\leqnat \lfloorx\rfloor.ln p/p)-\operatorname{ln}(nat\lfloorx\rfloor)-(lnx-\operatorname{ln}(na
\lfloorx\rfloor))|
    by simp
    also have \ldots. \leq |(\sump|prime p ^p\leqnat \lfloorx\rfloor.ln p/p)-\operatorname{ln}(nat \lfloorx\rfloor)|+|ln
x-ln (nat \lfloorx\rfloor)|
    by (rule abs-triangle-ineq4)
    also have |(\sump|prime p^p\leqnat \lfloorx\rfloor. ln p/p) - ln (nat \lfloorx\rfloor)|\leq7
        using 1 by (intro Mertens) auto
    also have |ln x - ln (nat \lfloorx\rfloor)| = ln x - ln (nat \lfloorx\rfloor)
        using 1 by (intro abs-of-nonneg) auto
    also from 1 have ... \leq (x - nat \lfloorx\rfloor) / nat \lfloorx\rfloor by (intro ln-diff-le) auto
    also have \ldots\leq(x-nat \lfloorx\rfloor) / 1 using 1 by (intro divide-left-mono) auto
    also have \ldots= .. x - nat \lfloorx\rfloor by simp
    also have \ldots. \1 using 1 by linarith
    also have (\sump|prime p\wedgep\leqnat \lfloorx\rfloor. ln p/p)=(\sump| prime p}\wedge\mathrm{ real ps
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

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

for a non-principal character $\chi$ and show that they are, in fact, bounded, which is ultimately a consequence of the non-vanishing of $L(1, \chi)$ for nonprincipal $\chi$.
context dcharacter
begin
context
includes no-vec-lambda-notation 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 $x: x>0$
shows norm $(L *$ sum-upto $(\lambda k . \chi k *$ mangoldt $k / k) x-$ sum-upto $(\lambda k . \chi k$ * $\ln k / k) x$ )

$$
\leq 9 / 2 * \text { real }(\text { totient } n)
$$

proof -
define $B$ where $B=3 *$ real (totient $n$ )
have sum-upto ( $\lambda k . \chi k * \ln k / k) x=$ sum-upto $\left(\lambda k . \chi k *\left(\sum d \mid d d v d k\right.\right.$. mangoldt d) / k) $x$
by (intro sum-upto-cong) (simp-all add: mangoldt-sum)
also have $\ldots=\operatorname{sum}$-upto $\left(\lambda k . \sum d \mid d d v d k . \chi k *\right.$ mangoldt $\left.d / k\right) x$
by (simp add: sum-distrib-left sum-distrib-right sum-divide-distrib)
also have $\ldots=$ sum-upto ( $\lambda k$. sum-upto $(\lambda d . \chi(d * k) *$ mangoldt $k /(d * k))$ $(x /$ real $k)) x$
by (rule sum-upto-sum-divisors)
also have $\ldots=$ sum-upto $(\lambda k . \chi k *$ mangoldt $k / k *$ sum-upto $(\lambda d . \chi d / d)$ $(x /$ real k)) $x$
by (simp add: sum-upto-def sum-distrib-left mult-ac)
also have $L *$ sum-upto $(\lambda k . \chi k *$ mangoldt $k / k) x-\ldots=$
sum-upto $(\lambda k .(L-$ sum-upto $(\lambda d . \chi d / d)(x /$ real $k)) *(\chi k *$
mangoldt $k / k)$ ) $x$
unfolding ring-distribs by (simp add: sum-upto-def sum-subtractf sum-distrib-left mult-ac)
also have norm $\ldots \leq$ sum-upto $(\lambda k . B / x *$ mangoldt $k$ ) $x$ unfolding sum-upto-def proof (rule sum-norm-le, goal-cases)
case (1k)
have norm $((L-$ sum-upto $(\lambda d . \chi d /$ of-nat $d)(x / k)) * \chi k *$ (mangoldt $k$ / of-nat k)) $\leq$ $(B *$ 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 $1 x / k] 1 x$
by (simp add: powr-minus $B$-def $L$-def divide-simps norm-minus-commute)
qed (insert 1, auto intro!: divide-nonneg-pos mangoldt-nonneg norm-le-1 simp: B-def)
also have $\ldots=B / x *$ mangoldt $k$ using 1 by simp
finally show? ?ase 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 $\ldots=9 / 2 *$ 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 $k / k) x-\operatorname{sum}$-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 $\chi$ )) 1 using conv-abscissa-le-O[OF nonprincipal]
by (intro fds-converges-deriv) (auto intro: le-less-trans)
hence summable $(\lambda n$. $-(\chi n * \ln n / n))$
by (auto simp: $f d s$-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-upto [OF this] obtain $c$ where
(sum-upto $(\lambda n . \chi n * \ln n / n) \longrightarrow c$ ) at-top by blast
hence $*:$ sum-upto $(\lambda k . \chi k * \ln k / k) \in O(\lambda-.1)$ 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-notation dcharacter-syntax
begin
lemma principal-dchar-sum-bound:
$\left(\lambda x .\left(\sum p \mid\right.\right.$ prime $p \wedge$ real $\left.\left.p \leq x . \chi_{0} p *(\ln p / p)\right)-\ln x\right) \in O(\lambda-.1)$
proof -
have fin [simp]: finite $\{p$. prime $p \wedge$ real $p \leq x \wedge Q p\}$ for $Q x$
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 $p \wedge$ real $p \leq x\}$ for $x$ by (simp del: fin)
define $P$ :: complex where $P=\left(\sum p \mid\right.$ prime $p \wedge p d v d n$. of-real $\left.(\ln p / p)\right)$
have $\left(\lambda x .\left(\sum p \mid\right.\right.$ prime $p \wedge$ real $\left.\left.p \leq x . \chi_{0} p *(\ln p / p)\right)-\ln x\right) \in$
$\Theta\left(\lambda x\right.$. of-real $\left(\left(\sum p \mid\right.\right.$ prime $p \wedge$ real $\left.\left.\left.p \leq x . \ln p / p\right)-\ln x\right)-P\right)($ is $-\epsilon$
$\Theta(? f))$
proof (intro bigthetaI-cong eventually-mono[OF eventually-gt-at-top[of real n]], goal-cases)
case (1 $x$ )
have $*:\{p$. prime $p \wedge$ real $p \leq x\}=$
$\{p$. prime $p \wedge$ real $p \leq x \wedge p$ dvd $n\} \cup\{p$. prime $p \wedge$ real $p \leq x \wedge \neg p$ $d v d n\}$
(is - = ? $A \cup$ ? $B$ ) by auto
have $\left(\sum p \mid\right.$ prime $p \wedge$ real $p \leq x$. of-real $\left.(\ln p / p)\right)=$
$\left(\sum p \in ? A\right.$. of-real $\left.(\ln p / p)\right)+\left(\sum p \in ? B\right.$. complex-of-real $\left.(\ln p / p)\right)$
by (subst $*$, subst sum.union-disjoint) auto
also from 1 have $? A=\{p$. prime $p \wedge p d v d n\}$ using $n$ by (auto dest: dvd-imp-le)
also have $\left(\sum p \in \ldots\right.$ of-real $\left.(\ln p / p)\right)=P$ by $(\operatorname{simp}$ add: $P$-def $)$
also have $\left(\sum p \in\right.$ ? B. of-real $\left.(\ln p / p)\right)=$
$\left(\sum p \mid\right.$ prime $p \wedge$ real $\left.p \leq x . \chi_{0} p *(\ln p / p)\right)$
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)
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 ${ }^{\prime}$ :
( $\lambda x$. sum-upto $\left(\lambda k . \chi_{0} k *\right.$ mangoldt $\left.\left.k / k\right) x-\operatorname{Ln} x\right) \in O(\lambda-.1)$
proof -
have $\left(\lambda x\right.$. sum-upto $\left(\lambda k . \chi_{0} k *\right.$ mangoldt $\left.k / k\right) x-$
$\left(\sum p \mid\right.$ prime $p \wedge$ real $\left.\left.p \leq x . \chi_{0} p *(\ln p / p)\right)\right) \in O(\lambda-.1)$
proof (intro bigoI[of-3] eventually-mono[OF eventually-gt-at-top[of 0]], goal-cases)
case (1 $x$ )
have norm (complex-of-real $\left(\sum k \mid\right.$ real $k \leq x \wedge$ coprime $k n$. mangoldt $\left.k / k\right)$
of-real $\left(\sum p \mid\right.$ prime $p \wedge p \in\{k$. real $k \leq x \wedge$ coprime $k n\} . \ln p /$
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 intro!: Nat.gr0I)
also have complex-of-real $\left(\sum k \mid\right.$ real $k \leq x \wedge$ coprime $k n$. mangoldt $\left.k / k\right)=$ sum-upto $\left(\lambda k . \chi_{0} k *\right.$ mangoldt $\left.k / k\right) x$
unfolding sum-upto-def of-real-sum using $n$
by (intro sum.mono-neutral-cong-left) (auto simp: principal-dchar-def intro!:

## Nat.gr0I)

also have complex-of-real $\left(\sum p \mid\right.$ prime $p \wedge p \in\{k$. real $k \leq x \wedge$ coprime $k n\}$. $\ln p / p)=$
$\left(\sum p \mid\right.$ prime $p \wedge$ real $\left.p \leq x . \chi_{0} p *(\ln p / p)\right)$
unfolding of-real-sum
by (intro sum.mono-neutral-cong-left)
(auto simp: principal-dchar-def le-nat-iff le-floor-iff prime-gt-0-nat intro!: finite-subset $[$ of - \{..nat $\lfloor x\rfloor\}]$ )
finally show ?case by simp
qed
from sum-in-bigo(1)[OF principal-dchar-sum-bound this] show ?thesis by $\operatorname{simp}$
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 \(\left(\lambda x .\left(\sum p \mid \operatorname{prime} p \wedge[p=h](\bmod n) \wedge \operatorname{real} p \leq x \ln p / p\right)-\ln x /\right.\)
totient \(n\) )
            \(\in O(\lambda-.1)(\) is \((\lambda x . \operatorname{sum}-(? A x)--) \in-)\)
proof -
    from assms obtain \(h^{\prime}\) where \(h^{\prime}:\left[h * h^{\prime}=\operatorname{Suc} 0\right](\bmod n)\)
    by (subst (asm) coprime-iff-invertible-nat) blast
    let ? \(A^{\prime}=\lambda x\). \(\{p . p>0 \wedge\) real \(p \leq x \wedge[p=h](\bmod n)\}\)
    let \(? B=\) dcharacters \(n-\left\{\chi_{0}\right\}\)
    have \([\) simp \(]: \chi_{0} \in\) dcharacters \(n\)
        by (auto simp: dcharacters-def principal.dcharacter-axioms)
    have \((\lambda x\). of-nat (totient \(n) *\left(\sum p \in ? A^{\prime} x\right.\). mangoldt \(\left.\left.p / p\right)-\ln x\right) \in\)
                        \(\Theta\left(\lambda x\right.\). sum-upto \(\left(\lambda k\right.\). \(\chi_{0} k *\) mangoldt \(\left.k / k\right) x-\ln x+\)
                            ( \(\sum \chi \in ? B . \chi h^{\prime} *\) sum-upto \((\lambda k . \chi k *\) mangoldt \(\left.k / k) x\right)\) )
    (is \(-\in \Theta(? f))\)
    proof (intro bigthetaI-cong eventually-mono[OF eventually-gt-at-top[of 0]], goal-cases)
    case ( \(1 x\) )
    have of-nat (totient \(n) *\left(\sum p \in ? A^{\prime} x\right.\). of-real (mangoldt \(\left.p / p\right)::\) complex \()=\)
                \(\left(\sum p \in ? A^{\prime} x\right.\). of-nat (totient \(\left.n\right) *\) of-real (mangoldt \(\left.p / p\right)\) )
            by (subst sum-distrib-left) simp-all
    also have \(\ldots=\) sum-upto \(\left(\lambda k . \sum \chi \in d\right.\) characters \(n . \chi\left(h^{\prime} * k\right) *(\) mangoldt \(k /\)
k)) \(x\)
            unfolding sum-upto-def
    proof (intro sum.mono-neutral-cong-left ballI, goal-cases)
            case (3p)
            have \(\left[h^{\prime} * p \neq 1\right](\bmod n)\)
            proof
                assume \(\left[h^{\prime} * p=1\right](\bmod n)\)
```

```
            hence [h*( }\mp@subsup{h}{}{\prime}*p)=h*1] (mod n) by (rule cong-scalar-left
            hence [h=h* h'*p] (mod n) by (simp add: mult-ac cong-sym)
            also have [h* h'*p=1*p](\operatorname{mod}n)
            using h' by (intro cong-scalar-right) auto
            finally have [p=h] (mod 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])
    next
        case (4 p)
        hence [p*\mp@subsup{h}{}{\prime}=h*h] (mod n) by (intro cong-scalar-right) auto
        also have [h* h'=1] (\operatorname{mod}n) using h' by simp
        finally have [h'* p=1] (mod n) by (simp add: mult-ac)
        thus ?case using h'4
        by (auto simp: sum-dcharacters sum-divide-distrib [symmetric] sum-distrib-right
[symmetric])
    qed auto
    also have ... = (\sum\chi\indcharacters n. sum-upto ( }\lambdak.\chi(\mp@subsup{h}{}{\prime}*k)* (mangoldt 
/ k)) x)
            unfolding sum-upto-def by (rule sum.swap)
    also have ... = (\sum\chi\indcharacters n. \chi h'* sum-upto ( }\lambdak.\chik*\mathrm{ (mangoldt }
/ 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 ( }\lambdak.\mp@subsup{\chi}{0}{}k*(\mathrm{ mangoldt k/k)) x +
                (\sum\chi\in?B.\chi \mp@subsup{h}{}{\prime}* sum-upto (\lambdak.\chik*(mangoldt k/k)) x)
            by (subst sum.remove [symmetric]) (auto simp: sum-distrib-left)
    also have coprime h' n
        using h' by (subst coprime-iff-invertible-nat, subst (asm) mult.commute) auto
    hence }\mp@subsup{\chi}{0}{}\mp@subsup{h}{}{\prime}=
        by (simp add: principal-dchar-def)
    finally show ?case using n 1 by (simp add: Ln-of-real)
qed
also have ?f }\inO(\lambda\mathrm{ -. 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
next
    case (2 \chi)
    then interpret dcharacter n G \chi by (simp-all add: G-def dcharacters-def)
    from 2 have \chi \not= \chio by auto
    thus ?case unfolding of-real-1
        by (intro landau-o.big.mult-in-1 nonprincipal-mangoldt-bound) auto
    qed
    finally have *: ( }\lambdax\mathrm{ . real (totient n)* ( }\sump\in?\mp@subsup{A}{}{\prime}x.mangoldt p / p) - ln x)
O(\lambda-. 1)
```

by (subst (asm) landau-o.big.of-real-iff)
have $\left(\lambda x\right.$. real $($ totient $n) *\left(\left(\sum p \in ? A x . \ln p / p\right)-\left(\sum p \in ? A^{\prime} x\right.\right.$. mangoldt $p /$ $p))) \in O(\lambda-.1)$
proof (intro landau-o.big.mult-in-1)
show $\left(\lambda x .\left(\sum p \in ? A x . \ln p / p\right)-\left(\sum p \in ? A^{\prime} x\right.\right.$. mangoldt $\left.\left.p / p\right)\right) \in O(\lambda-.1)$
unfolding landau-o.big.of-real-iff
proof (intro bigoI [of - 3] eventually-mono[OF eventually-gt-at-top [of 0]], goal-cases) case (1x)
have $\mid\left(\sum p \in ? A^{\prime} x\right.$. mangoldt $\left.p / p\right)-\left(\sum p \mid\right.$ prime $\left.p \wedge p \in ? A^{\prime} x . \ln p / p\right) \mid$ $\leq 3$
by (rule Mertens-mangoldt-versus-ln[where $n=$ nat $\lfloor x\rfloor]$ ) (auto simp: le-nat-iff le-floor-iff)
also have $\left\{p\right.$. prime $\left.p \wedge p \in ? A^{\prime} x\right\}=$ ? $A x$ 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 $\left(\lambda x\right.$. totient $\left.n *\left(\sum p \in ? A x . \ln p / p\right)-\ln x\right) \in O(\lambda-.1)$
by (simp add: field-simps)
also have $\left(\lambda x\right.$. totient $\left.n *\left(\sum p \in ? A x \cdot \ln p / p\right)-\ln x\right)=$
$\left(\lambda x\right.$. totient $n *\left(\left(\sum p \in ? A x \cdot \ln p / p\right)-\ln x /\right.$ totient $\left.\left.n\right)\right)$
using $n$ by (intro ext) (auto simp: field-simps)
also have $\ldots \in O(\lambda-.1) \longleftrightarrow$ ?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 \(p \wedge[p=h](\bmod n)\}\)
proof
    assume finite \(\{p\). prime \(p \wedge[p=h](\bmod n)\}\)
    then obtain \(K\) where \(K:\{p\). prime \(p \wedge[p=h](\bmod n)\} \subseteq\{. .<K\}\)
            by (auto simp: finite-nat-iff-bounded)
    have eventually \(\left(\lambda x .\left(\sum p \mid\right.\right.\) prime \(\left.p \wedge[p=h](\bmod n) \wedge \operatorname{real} p \leq x \cdot \ln p / p\right)=\)
                        \(\left(\sum p \mid\right.\) prime \(\left.\left.p \wedge[p=h](\bmod n) \cdot \ln p / p\right)\right)\) at-top
        using eventually-ge-at-top[of real \(K\) ] by eventually-elim (intro sum.cong, use
\(K\) in auto)
    hence \(\left(\lambda x\right.\). \(\left(\sum p \mid\right.\) prime \(p \wedge[p=h](\bmod n) \wedge\) real \(\left.\left.p \leq x . \ln p / p\right)\right) \in\)
                \(\Theta\left(\lambda-.\left(\sum p \mid\right.\right.\) prime \(\left.\left.p \wedge[p=h](\bmod n) . \ln p / p\right)\right)\) by (intro bigthetaI-cong)
auto
    also have \(\left(\lambda-.\left(\sum p \mid\right.\right.\) prime \(\left.\left.p \wedge[p=h](\bmod n) . \ln p / p\right)\right) \in O(\lambda-.1)\) by simp
    finally have \(\left(\lambda x .\left(\sum p \mid\right.\right.\) prime \(p \wedge[p=h](\bmod n) \wedge\) real \(\left.\left.p \leq x \ln p / p\right)\right) \in\)
\(O(\lambda-.1)\).
    from sum-in-bigo(2)[OF this Dirichlet-strong[OF assms]] and \(n\) show False by
simp
```


## 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

[1] 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.

