The Jordan-Hölder Theorem

Jakob von Raumer

September 19, 2015

Abstract
This submission contains theories that lead to a formalization of the proof of the Jordan-Hölder theorem about composition series of finite groups. The theories formalize the notions of isomorphism classes of groups, simple groups, normal series, composition series, maximal normal subgroups. Furthermore, they provide proofs of the second isomorphism theorem for groups, the characterization theorem for maximal normal subgroups as well as many useful lemmas about normal subgroups and factor groups. The formalization is based on the work in my first AFP submission [vR14] while the proof of the Jordan-Hölder theorem itself is inspired by course notes of Stuart Rankin [Ran05].

Contents
1 The Second Isomorphism Theorem for Groups 2
  1.1 Preliminaries ........................................ 2
2 Preliminary lemmas .................................. 7
3 More Facts about Subgroups .................... 7
4 Facts about Normal Subgroups .................. 10
5 Flattening the type of group carriers ........ 18
6 Simple Groups ..................................... 21
7 Facts about maximal normal subgroups ........ 23
8 Normal series and Composition series ........ 26
  8.1 Preliminaries ..................................... 26
  8.2 Normal Series ................................... 27
  8.3 Composition Series ............................. 33
9 Isomorphism Classes of Groups ............. 46
theory SndIsomorphismGrp

imports
  ~~/src/HOL/Algebra/Coset
  ../../../Secondary-Sylow/SubgroupConjugation

begin

1 The Second Isomorphism Theorem for Groups

1.1 Preliminaries

lemma (in group) triv-subgroup:
  shows subgroup \{1\} G
unfolding subgroup-def by auto

lemma (in group) triv-normal-subgroup:
  shows \{1\} < G
unfolding normal-def normal-axioms-def l-coset-def r-coset-def
using is-group triv-subgroup by auto

lemma (in group) normal-restrict-supergroup:
  assumes SsubG: subgroup S G
  assumes Nnormal:N < G
  assumes N ⊆ S
  shows N < (G{carrier := S})
proof –
interpret Sgrp: group G{carrier := S} using SsubG by (rule subgroup-imp-group)
show ?thesis
proof (rule Sgrp.normalI)
  show subgroup N (G{carrier := S}) using assms is-group by (metis subgroup-subgroup-of-subset normal-inv-iff)
next
  from SsubG have S ⊆ carrier G by (rule subgroup-imp-subset)
  thus ∀x ∈ carrier (G{carrier := S}). N #> G{carrier := S} x = x <# G{carrier := S} N
  using Nnormal unfolding normal-def normal-axioms-def l-coset-def r-coset-def
by fastforce
qed
qed

As this is maybe the best place this fits in: Factorizing by the trivial subgroup
is an isomorphism.

lemma (in group) trivial-factor-iso:
  shows the-elem ∈ (G Mod \{1\}) ≃ G
proof –
  have group-hom G G (λx. x) unfolding group-hom-def group-hom-axioms-def
  hom-def using is-group by simp

2
moreover have \((\lambda \cdot x. \cdot x) . \text{carrier } G = \text{carrier } G\) by simp
moreover have kernel \(G : G (\lambda \cdot x) = \{1\}\) unfolding kernel-def by auto
ultimately show \(?thesis\) using group-hom.FactGroup-iso by force
qed

And the dual theorem to the previous one: Factorizing by the group itself gives the trivial group

\textbf{lemma (in \text{group})} self-factor-iso:

\(\text{shows } (\lambda \cdot X . \text{the-elem } (\lambda \cdot 1 . X)) \in (G \text{ Mod } (\text{carrier } G)) \cong G\)\(\text{|carrier := \{}1\}\)

\textbf{proof} –

\(\text{have group } (G\)\(\text{|carrier := \{}1\})\) by (metis subgroup-imp-group triv-subgroup)

\(\text{hence group-hom } G (G\)\(\text{|carrier := \{}1\})\) (\(\lambda \cdot 1\)) unfolding group-hom-def group-hom-axioms-def hom-def using is-group by auto

moreover have \((\lambda \cdot 1) . \text{carrier } G = \text{carrier } (G\)\(\text{|carrier := \{}1\})\) by auto

moreover have kernel \(G (G\)\(\text{|carrier := \{}1\})\) (\(\lambda \cdot 1\)) = \(\text{carrier } G\) unfolding kernel-def by auto

ultimately show \(?thesis\) using group-hom.FactGroup-iso by force
qed

This theory provides a proof of the second isomorphism theorems for groups. The theorems consist of several facts about normal subgroups.

The first lemma states that whenever we have a subgroup \(S\) and a normal subgroup \(H\) of a group \(G\), their intersection is normal in \(G\)

\textbf{locale} second-isomorphism-grp = normal +
\textbf{fixes} \(S::\text{a set}\)
\textbf{assumes} subgrpS:subgroup \(S\ G\)

\textbf{context} second-isomorphism-grp

\textbf{begin}

\textbf{interpretation} groupS: group \(G\)\(\text{|carrier := \{}1\})\)
\textbf{using} subgrpS by (metis subgroup-imp-group)

\textbf{lemma} normal-subgrp-intersection-normal:

\(\text{shows } S \cap H \triangleq (G\)\(\text{|carrier := \{}1\})\)

\textbf{proof}(auto simp: groupS.normal-inv-iff)

\(\text{from subgrpS is-subgroup have } \bigwedge x. x \in \{S, H\} \Rightarrow \text{subgroup } x G\) by auto

\(\text{hence subgroup } (\bigcap \{S, H\}) G\) using subgroups-Inter by blast

\(\text{hence subgroup } (S \cap H) G\) by auto

moreover have \(S \cap H \subseteq S\) by simp

ultimately show subgroup \((S \cap H) (G\)\(\text{|carrier := \{}1\})\) using is-group subgroup subgroup-of-subset subgrpS by metis

\textbf{next}

\(\text{fix } g \ h\)

\(\text{assume } g: g \in S\) and \(hH: h \in H\) and \(hS: h \in S\)\

\(\text{from } g \ hH \text{ subgrpS show } g \otimes h \otimes \text{inv}_{G\)\(\text{|carrier := \{}1\})} g \in H\) by (metis inv-op-closed2 subgroup.mem-carrier subgroup-inv-equality)

3
lemma normal-set-mult-subgroup:
  shows subgroup (H <#> S) G
proof (rule subgroupI)
  show H <#> S ⊆ carrier G by (metis setmult-subset-G subgroup-imp-subset subgroupI)
next
  have 1 ∈ H 1 ∈ S using is-subgroup subgroupS subgroup.one-closed by auto
  hence 1 ⊗ 1 ∈ H <#> S unfolding set-mult-def by blast
  thus H <#> S ≠ {} by auto
next
  fix g
  assume g: g ∈ H <#> S
  then obtain h s where h:h ∈ H and s:s ∈ S and ghs:g = h ⊗ s unfolding set-mult-def by auto
  hence s ∈ carrier G by (metis subgroup.mem-carrier subgroupS)
  with g ghs obtain h' where h':h' ∈ H and g = s ⊗ h' using coset-eq unfolding r-coset-def l-coset-def by auto
  with s have inv g = (inv h') ⊗ (inv s) by (metis inv-mult-group mem-carrier subgroupI)
moreover from h' s subgroupS have inv h' ∈ H inv s ∈ S using subgroup.m-inv-closed m-inv-closed by auto
  ultimately show inv g ∈ H <#> S unfolding set-mult-def by auto
next
  fix g g'
  assume g: g ∈ H <#> S and h:g' ∈ H <#> S
  then obtain h h' s s' where hh' ss':h ∈ H h':H ∈ H s ∈ S s' ∈ S and g = h ⊗ s
  and g' = h' ⊗ s' unfolding set-mult-def by auto
  hence g ⊗ g' = (h ⊗ s) ⊗ (h' ⊗ s') by metis
  also from hh' ss' have inG:h ∈ carrier G h' ∈ carrier G s ∈ carrier G s' ∈ carrier G using subgroupS mem-carrier subgroupI by force+
  hence (h ⊗ s) ⊗ (h' ⊗ s') = h ⊗ (s ⊗ h') ⊗ s' using m-assoc by auto
  also from hh' ss' have h'' where h'':h'' ∈ H and s ⊗ h' = h'' ⊗ s using coset-eq unfolding r-coset-def l-coset-def by fastforce
  hence h ⊗ (s ⊗ h') ⊗ s' = h ⊗ (h'' ⊗ s) ⊗ s' by simp
  also from h'' inG have ... = (h ⊗ h'') ⊗ (s ⊗ s') using m-assoc mem-carrier by auto
  finally have g ⊗ g' = h ⊗ h'' ⊗ (s ⊗ s').
moreover with h'' hh' ss' have ... ∈ H <#> S unfolding set-mult-def using subgroupS subgroupI by fastforce
  ultimately show g ⊗ g' ∈ H <#> S by simp
qed

lemma oneH:1 ∈ H by (metis is-subgroup subgroupI subgroupI)
lemma H-contained-in-set-mult:
  shows $H \subseteq H <\#> S$
proof auto
  have $1 \in S$ by (metis subgroup.one-closed subgrpS)
  fix $x$
  assume $x \in H$
  with $1 \in S$ unfolding set-mult-def by force
  with $x \in H <\#> S$ by (metis mem-carrier r-one)
qed

lemma S-contained-in-set-mult:
  shows $S \subseteq H <\#> S$
proof auto
  fix $s$
  assume $s \in S$
  with $1 \in S$ unfolding set-mult-def by force
  with $s \in H <\#> S$ by (metis subgroup.mem-carrier l-one)
qed

lemma normal-intersection-hom:
  shows $\text{group-hom}\ (G\ (|\text{carrier}:=S|)\ (G\ (|\text{carrier}:=H <\#> S|)\ \text{Mod}\ H)\ (\lambda}\ g.\ H <\#> g))$
proof (auto del: equalityI simp: group-hom-def group-axioms-def hom-def groupS.is-group)
  have gr: group $(G\ (|\text{carrier}:=H <\#> S|))$ by (metis normal-set-mult-subgroup subgroup-imp-group)
  moreover have $H \subseteq H <\#> S$ by (rule H-contained-in-set-mult)
  moreover have subgroup $(H <\#> S)\ G$ by (metis normal-set-mult-subgroup)
  ultimately have $H <\#> (G\ (|\text{carrier}:=H <\#> S|))$ using normal-restrict-supergroup
  with gr show group $((G\ (|\text{carrier}:=H <\#> S|)\ \text{Mod}\ H))$ by (metis normal.factorgroup-is-group)
next
  fix $g$
  assume $g \in S$
  with subgrpS have $1 \otimes g \in H <\#> S$ unfolding set-mult-def by fastforce
  with $g \in H <\#> S$ by (metis l-one subgroup.mem-carrier subgrpS)
  thus $H <\#> g \in \text{carrier} (\{(G\ (|\text{carrier}:=H <\#> S|)\ \text{Mod}\ H)\})$ unfolding FactGroup-def RCOSETS-def r-coset-def by auto
next
  fix $g\ g'$ xa
  assume $g: g \in S$ and $g': g' \in S$
  hence $(H <\#> g) <\#> (H <\#> g') = (g \otimes g')$ by (metis rcos-sum subgroup.mem-carrier subgrpS)
  thus $H <\#> (g \otimes g') = (H <\#> g) <\#> G\ (|\text{carrier}:=H <\#> S|)$ (H <\#> g') unfolding set-mult-def by auto
qed

lemma normal-intersection-hom-kernel:
shows kernel \((G\langle \text{carrier} := S\rangle) (\langle G\langle \text{carrier} := H <\#> S\rangle \rangle \text{ Mod } H) (\lambda g. H \#> g) = H \cap S\)

proof –
  
  have kernel \((G\langle \text{carrier} := S\rangle) (\langle G\langle \text{carrier} := H <\#> S\rangle \rangle \text{ Mod } H) (\lambda g. H \#> g)\)
  
  = \{ g \in S. H \#> g = 1 \langle G\langle \text{carrier} := H <\#> S\rangle \rangle \text{ Mod } H \} unfolding kernel-def by auto
  
  also have ... = \{ g \in S. H \#> g = H \} unfolding FactGroup-def by auto
  
  also have ... = \{ g \in S. g \in H \} by (metis coset-eq is-subgroup lecoset-join2 rcos-self subgroup.mem-carrier subgrpS)
  
  also have ... = H \cap S by auto
  
  finally show ?thesis.

qed

lemma normal-intersection-hom-surj:
  
  shows \((\lambda g. H \#> g) \cdot \text{carrier} \langle G\langle \text{carrier} := S\rangle \rangle = \text{carrier} \langle (G\langle \text{carrier} := H <\#> S\rangle) \rangle \text{ Mod } H)\)
  
proof auto
  
  fix \( g \)
  
  assume \( g \in S \)
  
  hence \( g \in H <\#> S \) using S-contained-in-set-mult by auto
  
  thus \( H \#> g \in \text{carrier} \langle (G\langle \text{carrier} := H <\#> S\rangle) \rangle \text{ Mod } H \) unfolding FactGroup-def RCOSETS-def r-coset-def by auto
  
next
  
  fix \( x \)
  
  assume \( x \in \text{carrier} \langle G\langle \text{carrier} := H <\#> S\rangle \rangle \text{ Mod } H \)
  
  then obtain \( h s \) where \( h : h \in H \) and \( s : s \in S \) and \( x = H \#> (h \otimes s) \)
  
  unfolding FactGroup-def RCOSETS-def r-coset-def set-mult-def by auto
  
  hence \( x = (H \#> h) \#> s \) by (metis h s coset-mult-assoc mem-carrier subgroup.mem-carrier subgrpS subset)
  
  also have ... = H \#> s by (metis h is-group rcos-const)
  
  finally have \( x = H \#> s \).
  
  with \( s \) show \( x \in \text{op} \#> H \cdot S \) by simp

qed

Finally we can prove the actual isomorphism theorem:

theorem normal-intersection-quotient-isom:
  
  shows \((\lambda X. \text{the-elem} \langle (\lambda g. H \#> g) \cdot X \rangle) \in \langle (G\langle \text{carrier} := S\rangle) \rangle \text{ Mod } (H \cap S)\)
  
  \( \cong \langle (G\langle \text{carrier} := H <\#> S\rangle) \rangle \text{ Mod } H \)
  

end

end

theory SubgroupsAndNormalSubgroups imports
2 Preliminary lemmas

A group of order 1 is always the trivial group.

lemma (in group) order-one-triv-ifff:
shows \((\text{order } G = 1) = (\text{carrier } G = \{1\})\)
proof
assume order: \(\text{order } G = 1\)
then obtain \(x\) where \(x:\text{carrier } G = \{x\}\) unfolding order-def by (auto simp add: card-Suc-eq)
hence \(1 = x\) using one-closed by auto
next
assume carrier \(G = \{1\}\)
thus \(\text{order } G = 1\) unfolding order-def by auto
qed

lemma (in group) finite-pos-order:
assumes finite: finite (carrier \(G\))
shows \(0 < \text{order } G\)
proof
− from one-closed finite show \(?\text{thesis}\) unfolding order-def by (metis card-gt-0-iff subgroup-nonempty subgroup-self)
qed

lemma iso-order-closed:
assumes \(\varphi \in G \cong H\)
shows \(\text{order } G = \text{order } H\)
using assms
unfolding order-def iso-def by (metis (no-types) bij-betw-same-card mem-Collect-eq)

3 More Facts about Subgroups

lemma (in subgroup) subgroup-of-restricted-group:
assumes subgroup \(U (G\{\text{carrier } := H]\})\)
shows \(U \subseteq H\)
using assms
unfolding subgroup-imp-subset by force

lemma (in subgroup) subgroup-of-subgroup:
assumes group \(G\)
assumes subgroup \(U (G\{\text{carrier } := H]\})\)
shows subgroup \(U \subseteq G\)
proof
from assms(2) have \( U \subseteq H \) by (rule subgroup-of-restricted-group)
thus \( U \subseteq \text{carrier } G \) by (auto simp:subset)
next
fix \( x \) \( y \)
have \( a: x \otimes y = x \otimes G(\text{carrier} := H) \ y \) by simp
assume \( x \in U \) \( y \in U \)
with assms a show \( x \otimes y \in U \) by (metis subgroup.m-closed)
next
have \( 1_{G(\text{carrier} := H)} = 1 \) by simp
with assms show \( 1 \in U \) by (metis subgroup.one-closed)
next
have subgroup \( H \) \( G \)
fix \( x \)
assume \( x \in U \)
with assms (2) have \( \text{inv } G(\text{carrier} := H) x \in U \) by (rule subgroup.m-inv-closed)
moreover from assms \( \langle x \in U \rangle \) have \( \text{inv } G(\text{carrier} := H) x \in H \) by (metis in-mono subgroup-of-restricted-group)
with assms (1) \( \langle \text{subgroup } H \ G \rangle \) have \( \text{inv } G(\text{carrier} := H) x = \text{inv } x \) by (rule group.subgroup-inv-equality)
ultimately show \( \text{inv } x \in U \) by simp
qed

Being a subgroup is preserved by surjective homomorphisms

\textbf{lemma (in subgroup) surj-hom-subgroup:}
\begin{itemize}
\item assumes \( \varphi: \text{group-hom } G \ F \ \varphi \)
\item assumes \( \varphi|\text{surj}: \varphi' (\text{carrier } G) = \text{carrier } F \)
\item shows subgroup (\( \varphi' H \) ) \( F \)
\end{itemize}
\textbf{proof}
\begin{itemize}
\item from \( \varphi\text{surj} \) show \( \text{img-subset}; \varphi' H \subseteq \text{carrier } F \)
unfolding iso-def bij_betw_def by auto
\item next
\item fix \( f \) \( f' \)
\item assume \( h:f \in \varphi' H \) and \( h':f' \in \varphi' H \)
\item with \( \varphi\text{surj} \) obtain \( g \) \( g' \) where \( g:g \in H f = \varphi g \) and \( g':g' \in H f' = \varphi g' \) by auto
\item hence \( g \otimes G g' \in H \) by (metis m-closed)
\item hence \( \varphi (g \otimes G g') \in \varphi' H \) by simp
\item with \( g \varphi' H \) show \( f \otimes P f' \in \varphi' H \)
using group-hom.hom_mult by fastforce
\item next
\item have \( \varphi 1 \in \varphi' H \) by auto
\item with \( \varphi\text{show} \) \( 1_F \in \varphi' H \) by (metis group-hom.hom_one)
\item next
\item fix \( f \)
\item assume \( f:f \in \varphi' H \)
\item then obtain \( g \) where \( g:g \in H f = \varphi g \) by auto
\item hence \( \text{inv } g \in H \) by auto
\item hence \( \varphi (\text{inv } g) \in \varphi' H \) by auto
\item with \( \varphi\text{subset} \) show \( \text{inv } f \in \varphi' H \)
using group-hom.hom_inv by fastforce
\item qed
... and thus of course by isomorphisms of groups.

**lemma** iso-subgroup:
- assumes groups: group G group F
- assumes HG: subgroup H G
- assumes ϕ: ϕ ∈ G ∼ F
- shows subgroup (ϕ ' H) F

**proof** –
- from groups ϕ have group-hom G F ϕ unfolding group-hom-def group-hom-axioms-def iso-def by auto
- moreover from ϕ have ϕ ' (carrier G) = carrier F unfolding iso-def bij-betw-def by simp
- moreover note HG
- ultimately show thesis by (metis subgroup.surj-hom-subgroup)

qed

An isomorphism restricts to an isomorphism of subgroups.

**lemma** iso-restrict:
- assumes groups: group G group F
- assumes HG: subgroup H G
- assumes ϕ: ϕ ∈ G ∼ F
- shows (restrict ϕ H) ∈ (G\{carrier := H\}) ∼ (F\{carrier := ϕ ' H\})

unfolding iso-def hom-def bij-betw-def inj-on-def

**proof** auto

fix g h

assume g ∈ H h ∈ H

hence g ∈ carrier G h ∈ carrier G by (metis HG subgroup.mem-carrier)+

thus ϕ (g ⊗ G h) = ϕ g ⊗ F ϕ h using ϕ unfolding iso-def hom-def by auto

next

fix g h

assume g ∈ H h ∈ H g ⊗ G h ∉ H

hence False using HG unfolding subgroup-def by auto

thus undefined = ϕ g ⊗ F ϕ h by auto

next

fix g h

assume g: g ∈ H and h: h ∈ H and eq: ϕ g = ϕ h

hence g ∈ carrier G h ∈ carrier G by (metis HG subgroup.mem-carrier)+

with eq show g = h using ϕ unfolding iso-def bij-betw-def inj-on-def by auto

qed

The intersection of two subgroups is, again, a subgroup

**lemma** (in group) subgroup-intersect:
- assumes subgroup H G
- assumes subgroup H' G
- shows subgroup (H ∩ H') G

using assms unfolding subgroup-def by auto
4 Facts about Normal Subgroups

lemma (in normal) is-normal:
shows $H \lhd G$
by (metis coset-eq is-subgroup normalI)

Being a normal subgroup is preserved by surjective homomorphisms.

lemma (in normal) surj-hom-normal-subgroup:
assumes $\varphi: \text{group-hom} \; G \to F$
assumes $\varphi$ surj: $(\text{carrier} \; G) = \text{carrier} \; F$
shows $(\varphi \cdot H) \lhd F$
proof (rule group.normalI)
from $\varphi$ show group $F$
unfolding group-hom-def group-hom-axioms-def by simp
next
from $\varphi$ surj show subgroup $(\varphi \cdot H) \subseteq F$
by (rule surj-hom-subgroup)
next
show $\forall x \in \text{carrier} \; F. \; \varphi \cdot H \cdot x = x \cdot \varphi \cdot H$
proof
fix $f$
assume $f: f \in \text{carrier} \; F$
with $\varphi$ surj obtain $g$ where $g \cdot g \in \text{carrier} \; G \; f = \varphi \; g$ by auto
hence $\varphi \cdot H \cdot g \cdot f = \varphi \cdot H \cdot g \cdot \varphi \cdot g$ by simp
also have $\ldots = (\lambda x. \; (\varphi \cdot x) \cdot \varphi \; g) \; \cdot \; H$
unfolding r-coset-def image-def by auto
also have $\ldots = (\lambda x. \; \varphi \; g \cdot (\varphi \cdot x)) \; \cdot \; H$
unfolding l-coset-def by auto
also have $\ldots = (\lambda x. \; (\varphi \; g) \cdot (\varphi \; x)) \; \cdot \; H$
unfolding subset $g \varphi$ group-hom.hom-mult by fastforce
also have $\ldots = (\lambda x. \; \varphi \; g \cdot (\varphi \cdot x)) \; \cdot \; H$
unfolding l-coset-def image-def by auto
also have $\ldots = f \cdot \varphi \cdot H$
unfolding subset $g \varphi$ group-hom.hom-mult by auto
finally show $\varphi \cdot H \cdot f = f \cdot \varphi \cdot H$.
qed

Being a normal subgroup is preserved by group isomorphisms.

lemma iso-normal-subgroup:
assumes groups:group $G$ group $F$
assumes $HG: H \lhd G$
assumes $\varphi: \varphi \in G \cong F$
shows $(\varphi \cdot H) \lhd F$
proof
from groups $\varphi$ have group-hom $G \to F$
unfolding group-hom-def group-hom-axioms-def
iso-def by auto
moreover from $\varphi$ have $\varphi \cdot (\text{carrier} \; G) = \text{carrier} \; F$
unfolding iso-def bij-betw-def
by simp
moreover note $HG$

ultimately show thesis using normal.surj-hom-normal-subgroup by metis

qed

The trivial subgroup is a subgroup:

lemma (in group) triv-subgroup:
  shows subgroup \{1\} G
unfolding subgroup-def by auto

The cardinality of the right cosets of the trivial subgroup is the cardinality of the group itself:

lemma (in group) card-rcosets-triv:
  assumes finite (carrier G)
  shows card (rcosets \{1\}) = order G
proof
  have subgroup \{1\} G by (rule triv-subgroup)
  with assms have card (rcosets \{1\}) * card \{1\} = order G by (rule lagrange)
  thus thesis by (auto simp: card-Suc-eq)
qed

The intersection of two normal subgroups is, again, a normal subgroup.

lemma (in group) normal-subgroup-intersect:
  assumes M ◁ G and N ◁ G
  shows M ∩ N ◁ G
using assms subgroup-intersect is-group normal-inv-iff by simp

The set product of two normal subgroups is a normal subgroup.

lemma (in group) setmult-lcos-assoc:
  \[ H \subseteq carrier G; K \subseteq carrier G; x \in carrier G \]
  \[ \implies (x <# H) <#> K = x <# (H <#> K) \]
  by (force simp add: l-coset-def set-mult-def m-assoc)

lemma (in group) normal-subgroup-set-mult-closed:
  assumes M ◁ G and N ◁ G
  shows M <#> N ◁ G
proof (rule normalI)
  from assms show subgroup (M <#> N) G
    using second-isomorphism-grp.normal-set-mult-subgroup normal-imp-subgroup
    unfolding second-isomorphism-grp-def second-isomorphism-grp-axioms-def by force
next
  show \( \forall x \in carrier G. M \ <#> N \ <#> x = x \ <#> (M \ <#> N) \)
proof
  fix x
  assume x:x ∈ carrier G
  have M <#> N <#> x = M <#> (N <#> x) by (metis assms(1,2) normal-inv-iff setmult-rcos-assoc subgroup-imp-subset x)
    also have \ldots = M <#> (x <# N) by (metis assms(2) normal.coset-eq x)
    also have \ldots = (M ∩ N) <#> N by (metis assms(1,2) normal-imp-subgroup rcos-assoc-lcos subgroup-imp-subset x)
also have \( \ldots = (x \# M) \# N \) by (metis assms(1) normal.coset-eq x)
also have \( \ldots = x \# (M \# N) \) by (metis assms(1,2) normal-imp-subgroup
setmult-icos-assoc subgroup-imp-subset x)

finally show \( M \# N \# x = x \# (M \# N) \).
qed

The following is a very basic lemma about subgroups: If restricting the
carrier of a group yields a group it’s a subgroup of the group we’ve started
with.

lemma (in group) restrict-group-imp-subgroup:
  assumes \( H \subseteq \text{carrier } G \) group \( (G\{\text{carrier} := H\}) \)
  shows subgroup \( H \) \( G \)
proof
  from assms(1) show \( H \subseteq \text{carrier } G \).
next
  fix \( x \ y \)
  assume \( x \in H \ y \in H \)
  hence \( x \in \text{carrier} (G\{\text{carrier} := H\}) \ y \in \text{carrier} (G\{\text{carrier} := H\}) \) by auto
with assms(2) show \( x \otimes y \in H \) using assms(2) group.is-monoid monoid.m-closed
by fastforce
next
  show \( 1 \in H \) using assms(2) group.is-monoid monoid.one-closed by fastforce
next
  fix \( x \)
  assume \( x \in H \)
  hence \( x \in \text{carrier} (G\{\text{carrier} := H\}) \) by auto
  hence \( \inv_{G\{\text{carrier} := H\}} x \in \text{carrier} (G\{\text{carrier} := H\}) \) using assms(2)
group.inv-closed by fastforce
  hence \( \inv_{G\{\text{carrier} := H\}} x \in \text{carrier } G \) using \( x \) assms(1) by auto
  moreover have \( \inv_{G\{\text{carrier} := H\}} x \otimes x = 1 \) using assms(2) group.l-inv x
by fastforce
  moreover have \( x \in \text{carrier } G \) using \( x \) assms(1) by auto
  ultimately have \( \inv_{G\{\text{carrier} := H\}} x = \inv x \) using inv-equality[symmetric]
by auto
  thus \( \inv x \in H \) using assms(2) group.inv-closed x by fastforce
qed

A subgroup relation survives factoring by a normal subgroup.

lemma (in group) normal-subgroup-factorize:
  assumes \( \langle N < G \text{ and } N \subseteq H \text{ and } \text{subgroup } H \ G \text{ \rangle \) \( (G \text{ Mod } N) \)
  shows subgroup \( \langle \text{rcosets } G\{\text{carrier} := H\} N \rangle \) \( (G \text{ Mod } N) \)
proof
  interpret \( G\text{ModN}: \text{group } G \text{ Mod } N \) using assms(1) by (rule normal.factorgroup-is-group)
  have \( \langle N < G\{\text{carrier} := H\} \text{ using } \text{assms by (metis normal-restrict-supergroup) \rangle \) \( \langle \text{normal-imp-subgroup} \rangle \)
  hence \( \text{grpHN}: \text{group } \langle G\{\text{carrier} := H\} \text{ Mod } N \rangle \) by (rule normal.factorgroup-is-group)
  have \( \langle op \# G\{\text{carrier} := H\} = (\lambda U K. (\bigcup_{h \in U} \bigcup_{k \in K} \{h \otimes G\{\text{carrier} := H\} k\})) \rangle \) using set-mult-def by metis
normal
by
force
by
RCOSETS-def
by
RCOSETS-def
by
RCOSETS-def
mal
unfolding
N
normal
−
proof
by
metis
lemma
(A normality relation survives factoring by a normal subgroup.

qed
A normality relation survives factoring by a normal subgroup.

lemma (in group) normality-factorization:

assumes NG:N ⊲ G and NH:N ⊆ H and HG:H ⊲ G

shows (rcosets G_;carrier := H) N) ⊲ (G Mod N)

proof –

from assms(1) interpret GModN: group G Mod N by (metis normal, factorgroup-is-group)

show ?thesis

proof (auto simp: GModN.normal-inv-iff)

from assms show subgroup (rcosets G_;carrier := H) N) (G Mod N) using

normal-imp-subgroup normal-subgroup-factorize by force

next

fix U V

assume U:U ∈ carrier (G Mod N) and V:V ∈ rcosets G_;carrier := H) N

then obtain g where g:g ∈ carrier G U = N #> g unfolding FactGroup-def

RCOSETS-def by auto

from V obtain h where h:h ∈ H V = N #> h unfolding FactGroup-def

RCOSETS-def r-coset-def by auto

hence hG:h ∈ carrier G using HG normal-imp-subgroup subgroup.mem-carrier

by force

hence ghG:g ⊗ h ∈ carrier G using g m-closed by auto

from g h have g ⊗ h ⊗ inv g ∈ H using HG normal-inv-iff by auto

moreover have U #> V #> inv G Mod N U = N #> (g ⊗ h ⊗ inv g)

proof –

from g U have inv G Mod N U = N #> inv g using NG normal.inv-FactGroup

normal.rcos-inv by fastforce

hence U #> V #> inv G Mod N U = (N #> g) #> (N #> h) #> (N #> inv g) using g h by simp

also have ... = N #> (g ⊗ h) #> (N #> inv g) using g hG NG normal.rcos-sum by force

also have ... = N #> (g ⊗ h ⊗ inv g) using g inv-closed ghG NG normal.rcos-sum by force

finally show ?thesis .

qed

ultimately show U #> V #> inv G Mod N U ∈ rcosets G_;carrier := H)

N unfolding RCOSETS-def r-coset-def by auto

13
Factoring by a normal subgroups yields the trivial group iff the subgroup is
the whole group.

**Lemma (in normal) fact-group-trivial-iff:**
Assumes finite (carrier G)
Shows (carrier (G Mod H) = {1 G Mod H}) = (H = carrier G)
Proof
Assume carrier (G Mod H) = {1 G Mod H}
Moreover with assms lagrange have order (G Mod H) * card H = order G
Unfolding FactGroup-def order-def using is-subgroup by force
Ultimately have card H = order G unfolding order-def by auto
Thus H = carrier G using subgroup-imp-subset is-subgroup assms card-subset-eq
Unfolding order-def
By metis
Next
From assms have ordergt0:order G > 0 unfolding order-def by (metis subgroup-finite-imp-card-positive subgroup-self)
Assume H = carrier G
Hence card H = order G unfolding order-def by simp
With assms is-subgroup lagrange have card (rcosets H) * order G = order G by metis
With ordergt0 have card (rcosets H) = 1 by (metis mult-eq-self-implies-10 mult.commute neq0-conv)
Hence order (G Mod H) = 1 unfolding order-def FactGroup-def by auto
Thus carrier (G Mod H) = {1 G Mod H} using factorgroup-is-group by (metis group.order-one-triv-iff)
Qed

Finite groups have finite quotients.

**Lemma (in normal) factgroup-finite:**
Assumes finite (carrier G)
Shows finite (rcosets H)
Using assms unfolding RCOSETS-def by auto

The union of all the cosets contained in a subgroup of a quotient group acts
as a representation for that subgroup.

**Lemma (in normal) factgroup-subgroup-union-char:**
Assumes subgroup A (G Mod H)
Shows (\( \bigcup A \)) = \{ x \in carrier G. H \#> x \in A \}
Proof
Show \( \bigcup A \subseteq \{ x \in carrier G. H \#> x \in A \} \)
Proof
Fix x
Assume x:x \in \( \bigcup A \)
Then obtain a where a:a \in A x \in a by auto
With assms have xx:x \in carrier G using subgroup-imp-subset unfolding FactGroup-def RCOSETS-def r-coset-def by force
from assms a obtain \( y \) where \( y \cdot y \in \text{carrier } G \) \( a = H \#> y \) using subgroup-imp-subset

unfolding FactGroup-def RCOSETS-def by force
  with a have \( x \in H \#> y \) by simp
  hence \( H \#> y = H \#> x \) using y is-subgroup repr-independence by auto
  with \( y(a) \cdot a(1) \) have \( H \#> x \in A \) by auto
  with \( xx \) show \( x \in \{ x \in \text{carrier } G. \ H \#> x \in A \} \) by simp
qed
next
show \( \{ x \in \text{carrier } G. \ H \#> x \in A \} \subseteq \bigcup A \)
proof
  fix \( x \)
  assume \( xx \cdot x \in \{ x \in \text{carrier } G. \ H \#> x \in A \} \)
  hence \( xx \cdot x \in \text{carrier } G \) \( H \#> x \in A \) by auto
  moreover have \( x \in H \#> x \) by (metis is-subgroup rcos-self xx(1))
  ultimately show \( x \in \bigcup A \) by auto
qed
qed

lemma (in normal) factgroup-subgroup-union-subgroup:
  assumes subgroup A (G Mod H)
  shows subgroup (\( \bigcup A \)) G
proof –
  have subgroup \( \{ x \in \text{carrier } G. \ H \#> x \in A \} \) G
proof
  show \( \{ x \in \text{carrier } G. \ H \#> x \in A \} \subseteq \text{carrier } G \) by auto
next
  fix \( x \ y \)
  assume \( x \in \{ x \in \text{carrier } G. \ H \#> x \in A \} \) and \( y \in \{ x \in \text{carrier } G. \ H \#> x \in A \} \)
  hence \( xx \cdot x \in \text{carrier } G \) \( H \#> x \in A \) and \( y \cdot y \in \text{carrier } G \) \( H \#> y \in A \) by auto
  hence \( xyG:xx \cdot y \in \text{carrier } G \) by (metis m-closed)
  from assms \( x \ y \) have \( (H \#> x) \#< (y \#> y) \in A \) using subgroup.m-closed
unfolding FactGroup-def by fastforce
  hence \( H \#> (x \cdot y) \in A \) by (metis rcos-sum x(1) y(1))
  with \( xyG \) show \( x \cdot y \in \{ x \in \text{carrier } G. \ H \#> x \in A \} \) by simp
next
  have \( H \#> 1 \in A \) using assms subgroup.one-closed unfolding FactGroup-def
  by (metis coset-mult-one monoid.select-cons(2) subset)
  with assms one-closed show \( 1 \in \{ x \in \text{carrier } G. \ H \#> x \in A \} \) by simp
next
  fix \( x \)
  assume \( x \in \{ x \in \text{carrier } G. \ H \#> x \in A \} \)
  hence \( xx \cdot x \in \text{carrier } G \) \( H \#> x \in A \) by auto
  hence \( invx:xx \cdot x \in \text{carrier } G \) using inv-closed by simp
  from assms \( x \) have \( set-inv (H \#> x) \in A \) using subgroup.m-inv-closed by
  (metis inv-FactGroup subgroup.mem-carrier)
  hence \( H \#> (invx \cdot x) \in A \) by (metis rcos-inv x(1))
  with \( invx \) show \( invx \cdot x \in \{ x \in \text{carrier } G. \ H \#> x \in A \} \) by simp
qed

with assms factgroup-subgroup-union-char show \textit{thesis} by auto

qed

lemma \textbf{(in normal) factgroup-subgroup-union-normal}:
  assumes $A \triangleleft (G \Mod H)$
  shows $\bigcup A \triangleleft G$

proof (auto
  from assms show subgroup \{$x \in \text{carrier } G. \ H \triangleright x \in A$\} $\triangleleft G$
  by (metis (full-types) factgroup-subgroup-union-char factgroup-subgroup-union-subgroup
  normal-imp-subgroup)
  unfolding normal-def normal-axioms-def
  have $\in\text{carrier } G$ by (rule is-group)
  next
  interpret Anormal: normal A (G Mod H) using assms by simp
  fix $x\ y$
  interpret A
  assume $x\ y$
  unfolding r-coset-def by auto
  have $x x' \in \text{carrier } G \ H \triangleright x' \in A$ by auto
  from x(1) have $H x \ H \triangleright x \in \text{carrier } (G \Mod H)$ unfolding FactGroup-def RCOSETS-def by force
  with $x'$ have $(\text{invg }G \Mod H (H \triangleright x)) \otimes G \Mod H (H \triangleright x') \otimes G \Mod H (H \triangleright x) \in A$ using Anormal.inv-op-closed1 by auto
  hence (set-inv (H \triangleright x)) \triangleleft (H \triangleright x') \triangleleft (H \triangleright x) \in A$ using inv-FactGroup Hx unfolding FactGroup-def by auto
  hence $(H \triangleright (\text{inv } x)) \triangleleft (H \triangleright x') \triangleleft (H \triangleright x) \in A$ using x(1) by (metis rcos-inv)
  hence $(H \triangleright (\text{invg } x \otimes x')) \triangleleft (H \triangleright x) \in A$ by (metis inv-closed rcos-sum x'(1) x(1))
  hence $H \triangleright (\text{invg } x \otimes x' \otimes x) \in A$ by (metis inv-closed m-closed rcos-sum x'(1) x(1))
  moreover have $\text{invg } x \otimes x' \otimes x \in \text{carrier } G$ using $x\ x'$ by (metis inv-closed m-closed)
  ultimately have $\text{invg } x \otimes x' \otimes x \in \{x \in \text{carrier } G. \ H \triangleright x \in A\}$ by auto
  hence xcoset:x $\otimes$ (invg x $\otimes$ x' $\otimes$ x) $\in x$ using \{$x \in \text{carrier } G. \ H \triangleright x \in A$\}
  unfolding l-coset-def using x(1) by auto
  have $x \otimes (\text{invg } x \otimes x' \otimes x) = (x \otimes \text{invg } x) \otimes x' \otimes x$ by (metis Units-eq Units-inv-Units m-assoc m-closed x'(1) x(1))
  also have \ldots = $x' \otimes x$ by (metis l-one r-inv x'(1) x(1))
  also have \ldots = $y$ by (metis y = x' $\otimes$ x)
  finally have $x \otimes (\text{invg } x \otimes x' \otimes x) = y$.
  with xcoset show $y \in x$ using \{$x \in \text{carrier } G. \ H \triangleright x \in A$\} by auto
next
  interpret Anormal: normal A (G Mod H) using assms by simp
  fix $x\ y$

16
assume \( x : x \in \text{carrier} \ G \) \( y \in x <\# \{ x \in \text{carrier} \ G. \ H > x \in A \} \)
then obtain \( x' \) where \( x' \in \{ x \in \text{carrier} \ G. \ H > x \in A \} \) \( y = x \otimes x' \)
unfolding \( l \text{-coset-def} \) by \text{auto}

hence \( x':x' \in \text{carrier} \ G \ H > x' \in A \) by \text{auto}

from \( x(1) \) have \( \text{inex:inv} \ x \in \text{carrier} \ G \) by (rule \text{inv-closed})

hence \( \text{Hinex} : H > (\text{inv} \ x) \in \text{carrier} \ (G \ Mod \ H) \)
unfolding \( \text{FactGroup-def} \ \text{RCOSETS-def} \) by \text{force}

with \( x' \) have \( (\text{inv} \ G \ Mod \ H (H > x)) \odot G \ Mod \ H (H > x') \odot G \ Mod \ H (H > x) \in A \) using \( \text{inex:Anormal.inv-op-closed} \) by \text{auto}

hence \( (\text{set-inv} (H > x)) <\# (H > x') <\# (H > x) \in A \)

using \( \text{inv:FactGroup Hinex} \) unfolding \( \text{FactGroup-def} \) by \text{auto}

hence \( (H > x) <\# (H > x') <\# (H > x) \in A \) using \( \text{inex by} \ (\text{metis rcos-inv}) \)

hence \( (H > x) <\# (H > x') <\# (H > x) \in A \) by (metis \text{inv-inv} \( x(1) \))

hence \( H > (x \otimes x') <\# (H > x) \in A \) by (metis \text{rcos-sum} \( x(1) \))

hence \( H > (x \otimes x') \odot x \in A \) by (metis \text{inv-closed} \( m\text{-closed} \) \text{rcos-sum} \( x(1) \))

moreover have \( x \otimes x' \otimes x \in \text{carrier} \ G \) \( x x' \) by (metis \text{inv-closed} \( m\text{-closed} \))

ultimately have \( x \otimes x' \otimes x \in \{ x \in \text{carrier} \ G. \ H > x \in A \} \) by \text{auto}

hence \( \text{xcoset} : (x \otimes x' \otimes x) \odot x \in \{ x \in \text{carrier} \ G. \ H > x \in A \} \) \( > x \)

unfolding \( r \text{-coset-def} \) using \( \text{inex by} \) \text{auto}

have \( (x \otimes x' \otimes x) \odot x = (x \otimes x') \odot (x \otimes x) \) by (metis \text{Units-eq} \( \text{Units-inv-Units} \) \( m\text{-assoc} \) \( m\text{-closed} \) \( x(1) \) \( x(1) \))

also have \( ... = x \otimes x' \) using \( x(1) \) \text{l-inv} \( x(1) \) \( m\text{-closed} \) \( \text{m-one} \) by \text{auto}

also have \( ... = y \) by (metis \( y = x \otimes x' \))

finally have \( x \otimes x' \otimes x \otimes x = y \),

with \( \text{xcoset show} y \in \{ x \in \text{carrier} \ G. \ H > x \in A \} \) \( > x \) by \text{auto}

qed

with \( \text{assms show} \ ?\text{thesis by} \ (\text{metis} (\text{full-types}) \ \text{factgroup-subgroup-union-char} \) \( \text{normal-imp-subgroup} \))

qed

lemma (in \text{normal}) \text{factgroup-subgroup-union-factor}:

assumes \( A : (G \ Mod \ H) \)

shows \( A = \text{rcosets}_{G\{\text{carrier} := \bigcup A\}} H \)

proof –

have \( A = \text{rcosets}_{G\{\text{carrier} := \{ x \in \text{carrier} \ G. \ H > x \in A \}\}} H \)
proof \text{auto}

fix \( U \)

assume \( U : U \in A \)

then obtain \( x' \) where \( x':x' \in \text{carrier} \ G \ U = H > x' \)
using \( \text{assms subgroup-imp-subset} \)

unfolding \( \text{FactGroup-def} \ \text{RCOSETS-def} \) by \text{force}

with \( U \) have \( H > x' \in A \) by \text{simp}

with \( x' \) show \( U \in \text{rcosets}_{G\{\text{carrier} := \{ x \in \text{carrier} \ G. \ H > x \in A \}\}} H \) \text{unfolding} \( \text{RCOSETS-def} \) \( r \text{-coset-def} \) by \text{auto}

next

17
5 Flattening the type of group carriers

Flattening here means to convert the type of group elements from 'a set to 'a. This is possible whenever the empty set is not an element of the group.

definition flatten where
flatten (G:('a set, 'b) monoid-scheme) rep = (carrier=(rep ' (carrier G)),
mult=(λ x y. rep ((the-inv-into (carrier G) rep x) ⊗ G (the-inv-into (carrier G) rep y))), one=rep 1G)

lemma flatten-set-group-hom:
assumes group:group G
assumes inj:inj-on rep (carrier G)
shows rep ∈ hom G (flatten G rep)
unfolding hom-def
proof auto
  fix g
  assume g:g ∈ carrier G
  thus rep g ∈ carrier (flatten G rep) unfolding flatten-def by auto
next
  fix g h
  assume g:g ∈ carrier G and h:h ∈ carrier G
  hence rep g ∈ carrier (flatten G rep) rep h ∈ carrier (flatten G rep) unfolding flatten-def by auto
  hence rep g ⊗ flatten G rep rep h
  = rep (the-inv-into (carrier G) rep (rep g) ⊗ G the-inv-into (carrier G) rep (rep h)) unfolding flatten-def by auto
  also have ... = rep (g ⊗ G h) using inj g h by (metis the-inv-into-f-f)
  finally show rep (g ⊗ G h) = rep g ⊗ flatten G rep rep h.
qed

lemma flatten-set-group:
assumes group:group G
assumes inj:inj-on rep (carrier G)
shows group (flatten G rep)
proof (rule groupI)
  fix x y
  assume x:x ∈ carrier (flatten G rep) and y:y ∈ carrier (flatten G rep)
def y ≡ the-inv-into (carrier G) rep x and h ≡ the-inv-into (carrier G) rep y

hence \( x \otimes_{G} y = \text{rep} (g \otimes_{G} h) \) unfolding flatten-def by auto
moreover from \( g \)-def \( h \)-def have \( g \in \text{carrier} \ G \ h \in \text{carrier} \ G \)
using \( \text{inj} \ x \ y \ \text{the-inv-into} \) unfolding flatten-def by (metis partial-object.select-cones(1)
subset-refl)+
hence \( g \otimes_{G} h \in \text{carrier} \ G \) by (metis group group.is-monoid monoid.m-closed)
hence \( \text{rep} (g \otimes_{G} h) \in \text{carrier} \ (\text{flatten} \ G \ rep) \) unfolding flatten-def by simp
ultimately show \( x \otimes_{\text{flatten} \ G \ rep} y \in \text{carrier} \ (\text{flatten} \ G \ rep) \) by simp
next
show \( 1_{\text{flatten} \ G \ rep} \in \text{carrier} \ (\text{flatten} \ G \ rep) \) unfolding flatten-def by (simp add: group group.is-monoid)
next
fix \( x \ y \ z \)
assume \( x : x \in \text{carrier} \ (\text{flatten} \ G \ rep) \) and \( y : y \in \text{carrier} \ (\text{flatten} \ G \ rep) \) and \( z : z \in \text{carrier} \ (\text{flatten} \ G \ rep) \)
def \( g \equiv \text{the-inv-into} \ (\text{carrier} \ G) \ rep \ x \) and \( h \equiv \text{the-inv-into} \ (\text{carrier} \ G) \ rep \ y \)
and \( k \equiv \text{the-inv-into} \ (\text{carrier} \ G) \ rep \ z \)
hence \( x \otimes_{\text{flatten} \ G \ rep} y \otimes_{\text{flatten} \ G \ rep} z = (\text{rep} (g \otimes_{G} h)) \otimes_{\text{flatten} \ G \ rep} z \)
unfolding flatten-def by auto
also have \( \ldots = \text{rep} (\text{the-inv-into} (\text{carrier} \ G) \ rep (\text{rep} (g \otimes_{G} h)) \otimes_{G} k) \) using k-def unfolding flatten-def by auto
also from \( g \)-def \( h \)-def \( k \)-def have \( ghk_{G} : g \in \text{carrier} \ G \ h \in \text{carrier} \ G \ k \in \text{carrier} \ G \)
using \( \text{inj} \ x \ y \ z \ \text{the-inv-into} \) unfolding flatten-def by fastforce+
hence \( ghk_{G} \otimes_{G} h \in \text{carrier} \ G \) and \( hk_{G} \otimes_{G} k \in \text{carrier} \ G \) by (metis group group.is-monoid monoid.m-closed)+
hence \( \text{rep} (\text{the-inv-into} (\text{carrier} \ G) \ rep (\text{rep} (g \otimes_{G} h)) \otimes_{G} k) = \text{rep} ((g \otimes_{G} h) \otimes_{G} k) \)
unfolding flatten-def using \( \text{inj} \ \text{the-inv-into-f-f} \) by fastforce
also have \( \ldots = x \otimes_{G} (h \otimes_{G} k) \) using group group.is-monoid ghkG monoid.m-assoc by fastforce
also have \( \ldots = x \otimes_{\text{flatten} \ G \ rep} (y \otimes_{\text{flatten} \ G \ rep} z) \) unfolding h-def k-def flatten-def using \( x \) \( y \) by force
finally show \( x \otimes_{\text{flatten} \ G \ rep} y \otimes_{\text{flatten} \ G \ rep} z = x \otimes_{\text{flatten} \ G \ rep} (y \otimes_{\text{flatten} \ G \ rep} z) \).
next
fix \( x \)
assume \( x : x \in \text{carrier} \ (\text{flatten} \ G \ rep) \)
def \( g \equiv \text{the-inv-into} \ (\text{carrier} \ G) \ rep \ x \)
hence \( 1_{G} : g \in \text{carrier} \ G \) using \( \text{inj} \ x \) unfolding flatten-def using the-inv-into into
by force
have \( 1_{G} \in (\text{carrier} \ G) \) by (simp add: group group.is-monoid)
hence \( \text{the-inv-into} \ (\text{carrier} \ G) \ rep (1_{\text{flatten} \ G \ rep} g) = 1_{G} \) unfolding flatten-def
using the-inv-into-f-f inj by force
hence \( 1_{\text{flatten} \ G \ rep} \otimes_{\text{flatten} \ G \ rep} x = \text{rep} (1_{G} \otimes_{G} g) \) unfolding flatten-def g-def by simp
also have \( \ldots = \text{rep} \ g \) using \( g \)-group by (metis group.group.is-monoid monoid.l-one)
also have \( \ldots = x \) unfolding g-def using inj \( f \)-the-inv-into-f unfolding
flatten-def by force
finally show $1_{\text{flatten } G \text{ rep}} \otimes \text{flatten } G \text{ rep } x = x$.
next from group inj have hom:rep $\in$ hom $G$ (flatten $G$ rep) using flatten-set-group-hom
by auto
fix $x$
assume $x:x \in$ carrier (flatten $G$ rep)
def $g \equiv \text{the-inv-into} (\text{carrier } G) \text{ rep } x$
hence $gG:g \in$ carrier $G$ using inj $x$
unfolding flatten-def
by force
hence $\text{inv}_G : g \in$ carrier $G$ by (metis group group.inv-closed)
moreover have $\text{rep } (\text{inv}_G g) \otimes \text{flatten } G \text{ rep } x = \text{rep } (\text{inv}_G g) \otimes \text{flatten } G \text{ rep } (\text{rep } g)$
unfolding $g$-def using $f$-the-inv-into $f$ inj $x$ unfolding flatten-def by fastforce
hence $\text{rep } (\text{inv}_G g) \otimes \text{flatten } G \text{ rep } x = \text{rep } 1_G$ using $\text{inv}_G gG$ by (metis group group.inv-closed)
hence $\text{rep } (\text{inv}_G g) \otimes \text{flatten } G \text{ rep } x = 1_{\text{flatten } G \text{ rep}}$ unfolding flatten-def by auto
ultimately show $\exists y \in$ carrier (flatten $G$ rep). $y \otimes \text{flatten } G \text{ rep } x = 1_{\text{flatten } G \text{ rep}}$
by auto
qed

lemma (in normal) flatten-set-group-mod-inj:
shows inj-on $(\lambda U. \text{SOME } g. \ g \in U) (\text{carrier } (G \text{ Mod } H))$
proof (rule inj-onI)
fix $U \ V$
assume $U : U : \in$ carrier $(G \text{ Mod } H)$ and $V : V : \in$ carrier $(G \text{ Mod } H)$
then obtain $g h$ where $g:U = H \#> g \in$ carrier $G$ and $h:V = H \#> h \in$ carrier $G$
unfolding FactGroup-def RCOSETS-def by auto
hence notempty:$U \neq \{\} \ V \neq \{\} \ by \ (\text{metis empty-iff is-subgroup rcos-self})+$
assume $(\text{SOME } g. \ g \in U) = (\text{SOME } g. \ g \in V)$
with notempty have $(\text{SOME } g. \ g \in U) \in U \cap V \ by \ (\text{metis IntI ex-in-conv someI})$
thus $U = V$ by (metis Int-iff $g h$ is-subgroup repr-independence)
qed

lemma (in normal) flatten-set-group-mod:
shows group $(\text{flatten } (G \text{ Mod } H) (\lambda U. \text{SOME } g. \ g \in U))$
using factorygroup-is-group flatten-set-group-mod-inj by (rule flatten-set-group)

lemma (in normal) flatten-set-group-mod-iso:
shows $(\lambda U. \text{SOME } g. \ g \in U) \in (G \text{ Mod } H) \cong (\text{flatten } (G \text{ Mod } H) (\lambda U. \text{SOME } g. \ g \in U))$
unfolding iso-def bij-betw-def
apply (auto)
apply (metis flatten-set-group-mod-inj factorgroup-is-group flatten-set-group-hom)
apply (rule flatten-set-group-mod-inj)
unfolding flatten-def apply (auto)
done
end

theory SimpleGroups
imports SubgroupsAndNormalSubgroups ..\Secondary-Sylow/SndSylow SndIsomorphismGrp
begin

6 Simple Groups

locale simple-group = group +
  assumes order-gt-one:order G > 1
  assumes no-real-normal-subgroup:∀H. H ≪ G ⟹ (H = carrier G ∨ H = {1})

lemma (in simple-group) is-simple-group: simple-group G by (rule simple-group-axioms)

Simple groups are non-trivial.

lemma (in simple-group) simple-not-triv: carrier G ≠ {1} using order-gt-one
unfolding order-def by auto

Every group of prime order is simple

lemma (in group) prime-order-simple:
  assumes prime:prime (order G)
  shows simple-group G
proof
  from prime show 1 < order G unfolding prime-def by auto
next
fix H
assume H ≪ G
hence HG:subgroup H G unfolding normal-def by simp
hence card H dvd order G by (rule card-subgrp-dvd)
with prime have card H = 1 ∨ card H = order G unfolding prime-def by simp
thus H = carrier G ∨ H = {1}
proof
  assume card H = 1
  moreover from HG have 1 ∈ H by (metis subgroup.one-closed)
  ultimately show ?thesis by (auto simp: card-Suc-eq)
next
assume card H = order G
moreover from HG have H ⊆ carrier G unfolding subgroup-def by simp
moreover from prime have card (carrier G) > 1 unfolding order-def prime-def..
  hence finite (carrier G) by (auto simp:card-ge-0-finite)
  ultimately show ?thesis unfolding order-def by (metis card-subset-eq)
  qed
  qed

Being simple is a property that is preserved by isomorphisms.

lemma (in simple-group) iso-simple:
  assumes H: group H
  assumes iso: \( \varphi \in G \cong H \)
  shows simple-group H
  unfolding simple-group-def simple-group-axioms-def using assms(1)
  proof (auto del: equalityI)
  from iso have order G = order H unfolding iso-def order-def by simp
  by auto
  with order-gt-one show Suc 0 < order H by simp
  next
  have inv-iso: (inv-into (carrier G) \varphi) \in H \cong G using iso by (rule iso-sym)
  fix N
  assume NH:N \unlhd H and Nneq1:N \neq \{1_H\}
  then interpret Nnormal: normal N H by simp
  def M = (inv-into (carrier G) \varphi) \cdot N
  hence MG:M \unlhd G using inv-iso NH H by (metis is-group iso-normal-subgroup)
  have surj: \varphi \cdot carrier G = carrier H using iso unfolding iso-def bij-btw-def by simp
  hence MN: \varphi \cdot M = N unfolding M-def using Nnormal.subset image-inv-into-cancel
  by metis
  moreover have M \neq \{1\}
  proof (rule notI)
    assume M = \{1\}
    hence \varphi \cdot M = \{\varphi 1\} by (metis (full-types) image-empty image-insert)
    hence M = \{1\} by (metis (lifting) Nnormal.is-subgroup MN calculation singleton-iff subgroup.one-closed)
    thus False using Nneq1 MN by simp
  qed
  hence M = carrier G using no-real-normal-subgroup MG by auto
  ultimately show N = carrier H using surj by simp
  qed

As a corollary of this: Factorizing a group by itself does not result in a simple group!

lemma (in group) self-factor-not-simple: \neg simple-group (G Mod (carrier G))
proof
  assume asm:simple-group (G Mod (carrier G))
  have group (G[carrier := \{1\}]) by (metis subgroup-imp-group triv-subgroup)
  with asm self-factor-iso simple-group.iso-simple have simple-group (G[carrier := \{1\}]) by auto
  thus False using simple-group.simple-not-triv by force
theory MaximalNormalSubgroups
imports
  SubgroupsAndNormalSubgroups
  SimpleGroups
begin

7 Facts about maximal normal subgroups

A maximal normal subgroup of $G$ is a normal subgroup which is not contained in other any proper normal subgroup of $G$.

locale max-normal-subgroup =
  assumes proper: $H \neq \text{carrier } G$
  assumes max-normal: $\forall J. J < G \implies J \neq H \implies J \neq \text{carrier } G \implies \neg (H \subseteq J)$

Another characterization of maximal normal subgroups: The factor group is simple.

theorem (in normal) max-normal-simple-quotient:
  assumes finite: $\text{finite } (\text{carrier } G)$
  shows max-normal-subgroup $H G = \text{simple-group } (G \text{ Mod } H)$
proof
  assume max-normal-subgroup $H G$
  then interpret maxH: $\text{max-normal-subgroup } H G$.
  show $\text{simple-group } (G \text{ Mod } H)$ unfolding simple-group-def simple-group-axioms-def
  apply (rule conjI)
  apply (rule factorgroup-is-group)
  proof (auto del: equalityI)
  from finite factgroup-finite factorgroup-is-group group finite-pos-order have gt0: $0 < \text{card } (\text{rcosets } H)$
    unfolding FactGroup-def order-def by force
  from maxH proper finite have $\text{carrier } (G \text{ Mod } H) \neq \{1_{G \text{ Mod } H}\}$ using fact-group-trivial-iff by auto
  hence $1 \neq \text{order } (G \text{ Mod } H)$ using factorgroup-is-group group.order-one-triv-iff
  by metis
  with gt0 show $\text{Suc } 0 < \text{order } (G \text{ Mod } H)$ unfolding order-def FactGroup-def by auto
next
  fix $A'$
  assume $A'_{\text{normal}}: A' < G \text{ Mod } H$ and $A'_{\text{nottriv}}: A' \neq \{1_{G \text{ Mod } H}\}$
  def $A \equiv \bigcup A'$
  have $A2: A < G$ using $A'_{\text{normal}}$ unfolding A-def by (rule factgroup-subgroup-union-normal)
  have $H \in A'$ using $A'_{\text{normal}}$ normal-imp-subgroup subgroup one-closed unfolding FactGroup-def by force

qed

end
hence \( H \subseteq A \) unfolding \( A \)-def by auto

hence \( A_1 : H \triangleleft (G\langle \text{carrier} := A \rangle) \) using \( A_2 \) is-normal by (metis is-subgroup maxH.max-normal normal-restrict-supergroup subgroup-self)

have \( A_2 : A' = \text{rcosets} G\langle \text{carrier} := A \rangle \) unfolding \( A \)-def using factgroup-subgroup-union-factor \( A' \) normal normal-imp-subgroup by auto

from \( A_2 \) interpret normalHA: normal \( H \) (\( G\langle \text{carrier} := A \rangle \)) by metis

have \( H \subseteq A \) using normalHA.is-subgroup subgroup-imp-subset by force

with \( A_2 \) have \( A = H \lor A = \text{carrier} G \) using maxH.max-normal by auto

thus \( A' = \text{carrier} (G \text{ Mod } H) \)

proof (rule disjE)

assume \( A = H \)

hence \( (G\langle \text{carrier} := A \rangle \text{ Mod } H) = \{1_{(G\langle \text{carrier} := A \rangle \text{ Mod } H)}\} \)

by (metis finite is-group normalHA.fact-group-trivial-iff normalHA.subgroup-self normalHA.subset subgroup-finite subgroup-of-restricted-group subgroup-of-subgroup subset-antisym)

also have \( \ldots = \{1_{G \text{ Mod } H} \) unfolding FactGroup-def by auto

finally have \( A' = \{1_{G \text{ Mod } H}\} \) using \( A_3 \) unfolding FactGroup-def by simp

with \( A \) 'nottriv' show \( \text{thesis} .. \)

next

assume \( A = \text{carrier} G \)

hence \( (G\langle \text{carrier} := A \rangle \text{ Mod } H) = G \text{ Mod } H \) by auto

thus \( A' = \text{carrier} (G \text{ Mod } H) \) using \( A_3 \) unfolding FactGroup-def by simp

qed

qed

next

assume simple: simple-group \( (G \text{ Mod } H) \)

show max-normal-subgroup \( H \ G \)

proof

from simple have carrier \( (G \text{ Mod } H) \neq \{1_{G \text{ Mod } H}\) unfolding simple-group-def simple-group-axioms-def order-def by auto

with finite fact-group-trivial-iff show \( H \neq \text{carrier} G \) by auto

next

fix \( A \)

assume \( A : A \triangleleft G \) \( A \neq H \) \( A \neq \text{carrier} G \)

show \( \neg H \subseteq A \)

proof

assume \( H A : H \subseteq A \)

hence \( H \triangleleft (G\langle \text{carrier} := A \rangle) \) by (metis \( A(1) \) inv-op-closed2 is-subgroup normal-inv-iff normal-restrict-supergroup)

then interpret normalHA: normal \( H \) (\( G\langle \text{carrier} := A \rangle \)) by simp

from finite have finiteA: finite \( A \) using \( A(1) \) normal-imp-subgroup by (metis subgroup-finite)

have \( \text{rcosets}(G\langle \text{carrier} := A \rangle) \) \( H \triangleleft G \text{ Mod } H \) using normality-factorization is-normal HA \( A(1) \) by auto

with simple have \( \text{rcosets}(G\langle \text{carrier} := A \rangle) \) \( H = \{1_{G \text{ Mod } H}\} \lor \text{rcosets}(G\langle \text{carrier} := A \rangle) \)

\( H = \) carrier \( (G \text{ Mod } H) \)

unfolding simple-group-def simple-group-axioms-def by auto

thus False

24
proof
  assume \[\text{rcosets } G | \text{carrier} := A] H = \{1 \text{Mod } H\}
  hence \[\text{rcosets } G | \text{carrier} := A] H = \{1G | \text{carrier} := A] \text{Mod } H\] unfolding
FactGroup-def
with finite A have H = A using normal HA fact-group-trivial-iff unfolding
FactGroup-def
with A(2) show \(?thesis\) by simp
next
  assume AHG\(\text{H} H G\): \[\text{rcosets } G | \text{carrier} := A] H = \text{carrier } (G \text{ Mod } H)
  have A = \text{carrier } G unfolding FactGroup-def RCOSETS-def
proof
  show \(A \subseteq \text{carrier } G\) using A(1) normal-imp-subgroup subgroup-imp-subset
  by metis
next
  show \(\text{carrier } G \subseteq A\)
  proof
    fix x
    assume x: x \in \text{carrier } G
    hence H \#> x \in \text{rcosets } H unfolding RCOSETS-def
    with AHG H have H \#> x \in \text{rcosets } G | \text{carrier} := A] H unfolding
FactGroup-def
then obtain x' where x' : x' \in A H \#> x = H \#> G | \text{carrier} := A] x'
unfolding RCOSETS-def
hence H \#> x = H \#> x' unfolding r-coset-def
hence x \in H \#> x' by (metis is-subgroup rcos-self x)
hence x \in A \#> x' using HA
unfolding r-coset-def
thus x \in A using x'(1) unfolding r-coset-def
using subgroup.m-closed
A(1) normal-imp-subgroup
by force
qed
qed
with A(3) show \(?thesis\) by simp
qed
qed
qed
end

theory CompositionSeries imports SimpleGroups MaximalNormalSubgroups begin
8 Normal series and Composition series

8.1 Preliminaries

A subgroup which is unique in cardinality is normal:

**lemma (in group) unique-sizes-subgrp-normal:**

assumes fin:finite (carrier G)
assumes ∃!Q. Q ∈ subgroups-of-size q
shows (THE Q. Q ∈ subgroups-of-size q) ≺ G

**proof**

from assms obtain Q where Q ∈ subgroups-of-size q by auto
def Q ≡ THE Q.
with assms have Qsize:Q ∈ subgroups-of-size q using theI by metis
hence QG:subgroup Q G and cardQ:card Q = q unfolding subgroups-of-size-def by auto
from QG have Q ≺ G apply(rule normalI)
proof
fix g
assume g:g ∈ carrier G
hence invg:inv g ∈ carrier G by (metis inv-closed)
with fin Qsize have conjugation-action q (inv g) Q ∈ subgroups-of-size q by (metis conjugation-is-size-invariant)
with g Qsize have (inv g)^# (Q ^# g) ∈ subgroups-of-size q unfolding conjugation-action-def by auto
with invg have inv g^# (Q^# g) = Q by (metis Qsize assms (2) inv-inv)
with QG QG g show Q ^# g = g ^# Q by (rule conj-wo-inv)
qed
with Q-def show ?thesis by simp
qed

A group whose order is the product of two distinct primes p and q where p < q has a unique subgroup of size q:

**lemma (in group) pq-order-unique-subgrp:**

assumes finite:finite (carrier G)
assumes orderG:order G = q * p
assumes primep:prime p and primeq:prime q and pq:p < q
shows ∃!Q. Q ∈ (subgroups-of-size q)

**proof**

from primep primeq pq have nqdvdp:- (q dvd p) by (metis less-not-refl3 prime-def)
def calM ≡ {s. s ⊆ carrier G ∧ card s = q ^ 1}
def RelM ≡ {(N1, N2). N1 ∈ calM ∧ N2 ∈ calM ∧ (∃g∈carrier G. N1 = N2 ^# g)}
interpret syl: snd-sylow G q 1 p calM RelM
unfolding snd-sylow-def sylow-def snd-sylow-axioms-def sylow-axioms-def using is-group primep orderG finite nqdvdp calM-def RelM-def by auto
obtain Q where Q:Q ∈ subgroups-of-size q by (metis (lifting, mono-tags) mem-Collect-eq power-one-right subgroups-of-size-def syl.sylow-thm)
thus ?thesis
proof (rule ex1I)
fix P
assume P:P ∈ subgroups-of-size q
have card (subgroups-of-size q) mod q = 1 by (metis power-one-right syl.p-sylow-mod-p)

moreover have card (subgroups-of-size q) dvd p by (metis power-one-right syl.num-sylow-dvd-remainder)
ultimately have card (subgroups-of-size q) = 1 using pq primep by (metis Divides.mod-less prime-def)

with Q P show P = Q by (auto simp:card-Suc-eq)
qed
qed

... And this unique subgroup is normal.
corollary (in group) pq-order-subgrp-normal:
assumes finite: finite (carrier G)
assumes orderG:order G = q * p
assumes primep:prime p and primeq:prime q and pq:p < q
shows (THE Q. Q ∈ subgroups-of-size q) ≲ G
using assms by (metis pq-order-unique-subgrp unique-sizes-subgrp-normal)

The trivial subgroup is normal in every group.
lemma (in group) trivial-subgroup-is-normal:
shows {1} ≲ G
unfolding normal-def normal-axioms-def r-coset-def l-coset-def by (auto intro: normalI subgroupI simp: is-group)

8.2 Normal Series

We define a normal series as a locale which fixes one group G and a list S of subsets of G’s carrier. This list must begin with the trivial subgroup, end with the carrier of the group itself and each of the list items must be a normal subgroup of its successor.

locale normal-series = group +
fixes S
assumes notempty:S ≠ []
assumes hd:hd S = {1}
assumes last:last S = carrier G
assumes normal:∀ i. i + 1 < length S ⇒ (S ! i) ≲ G[(carrier := S ! (i + 1))]

lemma (in normal-series) is-normal-series: normal-series G S by (rule normal-series-axioms)

For every group there is a "trivial" normal series consisting only of the group itself and its trivial subgroup.

lemma (in group) trivial-normal-series:
shows normal-series G {{1}, carrier G}
unfolding normal-series-def normal-series-axioms-def
using is-group trivial-subgroup-is-normal by auto
We can also show that the normal series presented above is the only such with a length of two:

**Lemma (in `normal-series`) length-two-unique:**

- **Assumes** `length \( G \) = 2`
- **Shows** `\( G = \{1\}, \text{carrier } G \)`

**Proof:** (rule `nth-equalityI`)
- **From** `assms` show `length \( G \) = length \( \{1\}, \text{carrier } G \)` by `auto`

**Next**
- **Show** `\( \forall i < length \ G. \ G \! i = \{1\}, \text{carrier } G \)`
  **Proof:** (rule `allI`, rule `implI`)
    - Fix `i`
    - Assume `i : i < length \( \ G \)`
      **With assms** have `i = 0 \lor i = 1` by `auto`
      **Thus** `\( G \! i = \{1\}, \text{carrier } G \)`
      **Proof:** (rule `disjE`)
        - Assume `i : i = 0`
          **With assms** have `\( G \! i = \text{hd } \ G \)` by (metis `hd-conv-nth notempty`)
          **Thus** `\( G \! i = \{1\}, \text{carrier } G \)`
          **Using** `hd i` by `simp`
        - Next
          - Assume `i : i = 1`
            **With assms** have `\( G \! i = \text{last } \ G \)` by (metis `diff-add-inverse last-conv-nth nat-1-add-1 notempty`)
            **Thus** `\( G \! i = \{1\}, \text{carrier } G \)`
            **Using** `last i` by `simp`
    - Qed
    - Qed
    - Qed

We can construct new normal series by expanding existing ones: If we append the carrier of a group \( G \) to a normal series for a normal subgroup \( H \triangleleft G \) we receive a normal series for \( G \).

**Lemma (in `group`) normal-series-extend:**

- **Assumes** `normal: normal-series \( G(\text{carrier := } H[]) \)`
- **Assumes** `\( H \triangleleft G \)`
- **Shows** `normal-series \( G(\text{H @ [carrier } G]) \)`

**Proof**
- From `normal interpret` `normalH: normal-series \( G(\text{carrier := } H[]) \)`.
  - From `normalH.hd have` `hd \( H \) = \{1\}` by `simp`
    **With** `normalH.notempty have` `hdTriv:hd \( \{1\} @ [carrier } G]) = \{1\}` by (metis `hd-append2`)
  - **Show** `\?thesis unfolding` `normal-series-def normal-series-axioms-def using` `is-group`
    **Proof** `auto`
      - Fix `x`
        **Assume** `x \in hd \( \{1\} @ [carrier } G])`
        **With** `hdTriv` show `x = 1` by `simp`
    - Next
      - From `hdTriv show` `1 \in hd \( \{1\} @ [carrier } G])` by `simp`
    - Next
      - Fix `i`
        **Assume** `i : i < length \( \{1\} \)`

28
show $(\mathcal{S} @ [\text{carrier } G]) ! i \triangleleft G(\text{carrier } := (\mathcal{S} @ [\text{carrier } G]) ! \text{Suc } i)$

proof (cases $i + 1 < \text{length } \mathcal{S}$)
  case True
    with normalH.normal have $\mathcal{S} ! i \triangleleft G(\text{carrier } := \mathcal{S} ! (i + 1))$ by auto
    with i have $(\mathcal{S} @ [\text{carrier } G]) ! i \triangleleft G(\text{carrier } := \mathcal{S} ! (i + 1))$ using nth-append by metis
    with True show $(\mathcal{S} @ [\text{carrier } G]) ! i \triangleleft G(\text{carrier } := (\mathcal{S} @ [\text{carrier } G]) ! \text{Suc } i)$ using nth-append Suc-eq-plus1 by metis
  next
    case False
    with i have $i : i + 1 = \text{length } \mathcal{S}$ by simp
    from i have $(\mathcal{S} @ [\text{carrier } G]) ! i = \mathcal{S} ! i$ by (metis nth-append)
    also from $i \geq 2$ normalH.normal have ... = last $\mathcal{S}$ by (metis add-diff-cancel-right' last-conv-nth)
    also from normalH.normal have ... = $H$ by simp
    finally have $(\mathcal{S} @ [\text{carrier } G]) ! i = H$.
    moreover from $i \geq 2$ have $(\mathcal{S} @ [\text{carrier } G]) ! (i + 1) = \text{carrier } G$ by (metis nth-append-length)
    ultimately show thesis using $H \cdot G$ by auto
  qed
  qed
  qed

All entries of a normal series for $G$ are subgroups of $G$.

lemma (in normal-series) normal-series-subgroups:
  shows $i < \text{length } \mathcal{S} \implies \text{subgroup } (\mathcal{S} ! i) G$
proof
  have $i + 1 < \text{length } \mathcal{S} \implies \text{subgroup } (\mathcal{S} ! i) G$
  proof (induction length $\mathcal{S} - (i + 2)$ arbitrary: i)
    case 0
    hence $i : i + 2 = \text{length } \mathcal{S}$ using assms by simp
    hence $i : i + 1 = \text{length } \mathcal{S} - 1$ using assms by force
    from $i \geq 2$ normal have $\mathcal{S} ! i \triangleleft G(\text{carrier } := \mathcal{S} ! (i + 1))$ by auto
    with $i \geq 2$ last notempty show subgroup $(\mathcal{S} ! i) G$ using last-conv-nth normal-imp-subgroup by fastforce
  next
    case (Suc k)
    from Suc(3) normal have $i : \text{subgroup } (\mathcal{S} ! i) (G(\text{carrier } := \mathcal{S} ! (i + 1)))$
    using normal-imp-subgroup by auto
    from Suc(2) have $k : k = \text{length } \mathcal{S} - ((i + 1) + 2)$ by arith
    with Suc have subgroup $(\mathcal{S} ! (i + 1)) G$ by simp
    with i show subgroup $(\mathcal{S} ! i) G$ by (metis is-group subgroup subgroup-of-subgroup)
    qed
    moreover have $i + 1 = \text{length } \mathcal{S} \implies \text{subgroup } (\mathcal{S} ! i) G$
    using last notempty last-conv-nth by (metis add-diff-cancel-right' subgroup-self)
    ultimately show $i < \text{length } \mathcal{S} \implies \text{subgroup } (\mathcal{S} ! i) G$ by force
    qed

The second to last entry of a normal series is a normal subgroup of $G$.  

29
lemma (in normal-series) normal-series-snd-to-last:
  shows $\mathcal{G} \ ! (\text{length } \mathcal{G} - 2) < G$
proof (cases $2 \leq \text{length } \mathcal{G}$)
  case False
  with notempty have length:length $\mathcal{G} = 1$ by (metis Suc-eq-plus1 leI length-0-conv less-2-cases plus-nat.add-0)
  with $\text{hd}$ have $\mathcal{G} ! (\text{length } \mathcal{G} - 2) = \{1\}$ using $\text{hd-conv-nth}$ notempty by auto
  with length show $\text{thesis}$ by (metis trivial-subgroup-is-normal)
next
  case True
  hence $(\text{length } \mathcal{G} - 2) + 1 < \text{length } \mathcal{G}$ by arith
  with normal last have $\mathcal{G} ! (\text{length } \mathcal{G} - 2) < G\{\text{carrier := } \mathcal{G} ! (\text{length } \mathcal{G} - 2) + 1\}\$ by auto
  have $1 + (1 + (\text{length } \mathcal{G} - (1 + 1))) = \text{length } \mathcal{G}$
  using True le-add-diff-inverse by presburger
  then have $\mathcal{G} ! (\text{length } \mathcal{G} - 2) < G\{\text{carrier := } \mathcal{G} ! (\text{length } \mathcal{G} - 1)\}$
  by (metis $\mathcal{G} ! (\text{length } \mathcal{G} - 2) < G \{\text{carrier := } \mathcal{G} ! (\text{length } \mathcal{G} - 2 + 1)\}$; 
  add.commute add-diff-cancel-left' one-add-one)
  with notempty last show $\text{thesis}$ using last-conv-nth by force
qed

Just like the expansion of normal series, every prefix of a normal series is again a normal series.

lemma (in normal-series) normal-series-prefix-closed:
  assumes $i \leq \text{length } \mathcal{G}$ and $0 < i$
  shows normal-series $(G\{\text{carrier := } \mathcal{G} ! (i - 1)\})\ (\text{take } i \mathcal{G})$
unfolding normal-series-def normal-series-axioms-def
using assms
apply (auto del)
apply (metis Suc-Suc diff-is-0-eq' gr_implies_not0 minus_n at diff_0 normal-series-subgroups not0_implies_Suc not_less_eq subgroup_imp_group zero_less_diff)
proof
  from assms have $\text{hd} \ (\text{take } i \mathcal{G}) = (\text{take } i \mathcal{G}) ! 0$ by (metis gr_implies_not0 $\text{hd-conv-nth}$ notempty take_eq Nil)
  also from assms have ... $= \mathcal{G} ! 0$ by (metis nth_take)
  also from $\text{hd}$ have ... $= \{1\}$ by (metis $\text{hd-conv-nth}$ notempty)
  finally show $\text{hd} \ (\text{take } i \mathcal{G}) = \{1\}$.
next
  from assms have last $\ (\text{take } i \mathcal{G}) = (\text{take } i \mathcal{G}) ! (\text{length } (\text{take } i \mathcal{G}) - 1)$ by (metis length_last_conv_n nth_eq_0_conv notempty take_eq Nil)
  also from assms have ... $= (\text{take } i \mathcal{G}) ! (i - 1)$ by (metis length_take min.absorb2)
  also from assms have ... $= \mathcal{G} ! (i - 1)$ by (metis diff_less nth_take zero_less_one)
  finally show last $\ (\text{take } i \mathcal{G}) = \mathcal{G} ! (i - \text{Suc } 0)$ by simp
next
  fix $j$
  assume $j: \text{Suc } j < i$
  hence $j + 1 < \text{length } \mathcal{G}$ using assms by simp
  with normal show $\mathcal{G} ! j < G\{\text{carrier := } \mathcal{G} ! (\text{Suc } j)\}$ by auto
qed

30
If a group’s order is the product of two distinct primes \( p \) and \( q \), where \( p < q \), we can construct a normal series using the only subgroup of size \( q \).

**Lemma (in group) pq-order-normal-series:**

- **Assumes** finite (carrier \( G \))
- **Assumes** order\( \): order \( G = q \cdot p \)
- **Assumes** prime\( \): prime \( p \) and prime\( \): prime \( q \) and \( pq < q \)
- **Shows** normal-series \( G \cdot \{1\}, (THE \ H. \ H \in \text{subgroups-of-size } q), \text{carrier } G \)

**Proof** –

- **Def** \( H \equiv (THE \ H. \ H \in \text{subgroups-of-size } q) \)
- **With** \( \text{assms} \Rightarrow H \cdot G \triangleleft G \)  by (metis \( pq \)-order-subgrp-normal)
- **Then interpret** group\( H \cdot G \cdot \{carrier := H\} \) unfolding normal-def by (metis subgroup-imp-group)
- **Have** normal-series \( (G[carrier := H]) \cdot \{1\}, H \) using group\( H \).trivial-normal-series

**QED**

The following defines the list of all quotient groups of the normal series:

**Definition (in normal-series) quotients**

- **Where** \( \text{quotients} = \text{map} (\lambda i. G[carrier := G ! (i + 1)] \mod G ! i) [0..<(\text{length } G - 1)] \)

The list of quotient groups has one less entry than the series itself:

**Lemma (in normal-series) quotients-length:**

- **Shows** length \( \text{quotients} + 1 = \text{length } G \)

**Proof** –

- **Have** length \( \text{quotients} + 1 = \text{length } [0..<(\text{length } G - 1)] + 1 \) unfolding quotients-def by simp
- **Also have** \( \ldots = (\text{length } G - 1) + 1 \) by (metis \( \text{diff-zero length-upt} \))
- **Also with** \( \text{notempty} \Rightarrow \text{have} \ldots = \text{length } G \)
  - by (simp add: ac-simps)
- **Finally show** \( ?\text{thesis} . \)

**QED**

**Lemma (in normal-series) last-quotient:**

- **Assumes** \( \text{length } G > 1 \)
- **Shows** last \( \text{quotients} = G \mod G ! (\text{length } G - 1 - 1) \)

**Proof** –

- **From** \( \text{assms} \Rightarrow \text{lsimp}:\text{length } G - 1 - 1 + 1 = \text{length } G - 1 \) by auto
- **From** \( \text{assms} \Rightarrow \text{have} \text{quotients} \neq [] \) unfolding quotients-def by auto
- **Hence** last \( \text{quotients} = \text{quotients} ! (\text{length } \text{quotients} - 1) \) by (metis \( \text{last-conv-nth} \))
- **Also have** \( \ldots = \text{quotients} ! (\text{length } G - 1 - 1) \) by (metis \( \text{add-diff-cancel-left} \) quotients-length add.commute)
- **Also have** \( \ldots = G[carrier := G ! ((\text{length } G - 1 - 1) + 1)] \mod G ! (\text{length } G - 1 - 1) \)
  - unfolding quotients-def using \( \text{assms} \) by auto
- **Also have** \( \ldots = G[carrier := G ! (\text{length } G - 1)] \mod G ! (\text{length } G - 1 - 1) \) using \( \text{lsimp} \) by simp
also have \( \ldots = G \text{ Mod } \mathfrak{G} \) ! \((\text{length } \mathfrak{G} - 1 - 1)\) using last last-conv-nth notempty

by force

finally show \( \text{thesis} \).

qed

The next lemma transports the constituting properties of a normal series along an isomorphism of groups.

lemma (in normal-series) normal-series-iso:

assumes \( H : \text{group } H \)

assumes \( \text{iso } \Psi \in G \cong H \)

shows normal-series \( H \) (map (image \( \Psi \)) \( \mathfrak{G} \))

apply (simp add: normal-series-def normal-series-axioms-def)

using \( H \) notempty apply simp

proof (rule conjI)

from \( H \) is-group iso have group-hom:group-hom \( G \ H \) \( \Psi \) unfolding group-hom-def

group-hom-axioms-def iso-def

have \( \text{hd} \) (map (image \( \Psi \)) \( \mathfrak{G} \)) = \( \Psi \cdot \{1\} \) by (metis \( \text{hd} \)-map \( \text{hd} \) notempty)

also have \( \ldots = \{\Psi \ 1\} \) by (metis image-empty image-insert)

also have \( \ldots = \{1_H\} \) using group-hom group-hom-hom-one by auto

finally show \( \text{hd} \) (map (op ' \( \Psi \)) \( \mathfrak{G} \)) = \( \{1_H\} \).

next

show last (map (op ' \( \Psi \)) \( \mathfrak{G} \)) = \( \text{carrier } H \land (\forall i. \text{Suc } i < \text{length } \mathfrak{G} \longrightarrow \Psi \cdot \mathfrak{G} !) \)

\( i < H \{\text{carrier} := \Psi \cdot \mathfrak{G} ! \text{ Suc } i)\}

proof (auto del: equalityI)

have last (map (op ' \( \Psi \)) \( \mathfrak{G} \)) = \( \Psi \cdot (\text{carrier } G) \) using last last-map notempty by metis

also have \( \ldots = \text{carrier } H \) using iso unfolding iso-def bij-betw-def by simp

finally show last (map (op ' \( \Psi \)) \( \mathfrak{G} \)) = \( \text{carrier } H \).

next

fix \( i \)

assume \( i : \text{Suc } i < \text{length } \mathfrak{G} \)

hence norm: \( \mathfrak{G} ! \) \( i < G\{\text{carrier} := \mathfrak{G} ! \text{ Suc } i)\} \) using normal by simp

moreover have \( \text{restrict } \Psi \ (\mathfrak{G} ! \text{ Suc } i) \in (G\{\text{carrier} := \mathfrak{G} ! \text{ Suc } i)\) \( \cong H\{\text{carrier} := \Psi \cdot \mathfrak{G} ! \text{ Suc } i)\) \)

by (metis \( H \) is-group iso iso-restrict normal-series-subgroups)

moreover have group (\( G\{\text{carrier} := \mathfrak{G} ! \text{ Suc } i)\}) by (metis i normal-series-subgroups subgroup-imp-group)

moreover hence \( \text{subgroup } (\mathfrak{G} ! \text{ Suc } i) \) \( G \) by (metis i normal-series-subgroups)

hence \( \text{subgroup } (\Psi \cdot \mathfrak{G} ! \text{ Suc } i) \) \( H \) by (metis \( H \) is-group iso iso-subgroup)

hence group \( (H\{\text{carrier} := \Psi \cdot \mathfrak{G} ! \text{ Suc } i)\}) \) by (metis \( H \) subgroup subgroup-is-group)

ultimately have \( \text{restrict } \Psi \ (\mathfrak{G} ! \text{ Suc } i) \cdot \mathfrak{G} ! \text{ Suc } i < H\{\text{carrier} := \Psi \cdot \mathfrak{G} ! \text{ Suc } i)\}

using is-group \( H \) iso-normal-subgroup by auto

moreover from \( \text{norm} \) have \( \mathfrak{G} ! \text{ Suc } i \subseteq \mathfrak{G} ! \text{ Suc } i \) unfolding normal-def subgroup-def by auto

hence \( \{y. \exists x \in \mathfrak{G} ! \text{ i. y} = (i f x \in \mathfrak{G} ! \text{ Suc } i \text{ then } \Psi x \text{ else undefined})\} = \{y. \exists x \in \mathfrak{G} ! \text{ i. y} = \Psi x \}\) by auto

ultimately show \( \Psi \cdot \mathfrak{G} ! \text{ Suc } i < H\{\text{carrier} := \Psi \cdot \mathfrak{G} ! \text{ Suc } i)\} \) unfolding restrict-def image-def by auto

qed

32
8.3 Composition Series

A composition series is a normal series where all consecutive factor groups are simple:

locale composition-series = normal-series +
  assumes simplefact:∀ i. i + 1 < length G ⇒ simple-group (G[carrier := G ! (i + 1)] Mod G ! i)

lemma (in composition-series) is-composition-series:
  shows composition-series G G
by (rule composition-series-axioms)

A composition series for a group G has length one if and only if G is the trivial group.

lemma (in composition-series) composition-series-length-one:
  shows (length G = 1) = (G = [\{1\}])
proof
  assume length G = 1
  with hd have length G = length [\{1\}] ∧ (∀ i < length G. G ! i = [\{1\}] ! i)
using hd-conv-nth notempty by force
  thus G = [\{1\}] using list-eq-iff-nth-eq by blast
next
  assume G = [\{1\}]
  thus length G = 1 by simp
qed

lemma (in composition-series) composition-series-triv-group:
  shows (carrier G = \{1\}) = (G = [\{1\}])
proof
  assume G:carrier G = \{1\}
  have length G = 1
  proof (rule ccontr)
    assume length G ≠ 1
    with notempty have length: length G ≥ 2 by (metis Suc-eq-plus1 length-0-conv less-2-cases not-less-plus-nat.add-0)
    with simplefact hd hd-conv-nth notempty have simple-group (G[carrier := G ! 1] Mod \{1\}) by force
    moreover have SG:subgroup (G ! 1) G using length normal-series-subgroups
    by auto
    hence group (G[carrier := G ! 1]) by (metis subgroup-imp-group)
    ultimately have simple-group (G[carrier := G ! 1]) using group.trivial-factor-iso simple-group.iso-simple by fastforce
    moreover from SG G have carrier (G[carrier := G ! 1]) = \{1\} unfolding subgroup-def by auto
    ultimately show False using simple-group.simple-not-triv by force
  qed
thus $\mathcal{G} = \{[1]\}$ by (metis composition-series-length-one)

next
  assume $\mathcal{G} = \{[1]\}$
  with last show $carrier\ G = \{1\}$ by auto
qed

The inner elements of a composition series may not consist of the trivial subgroup or the group itself.

lemma (in composition-series) inner-elements-not-triv:
  assumes $i + 1 < length\ \mathcal{G}$
  assumes $i > 0$
  shows $\mathcal{G} ! i \neq \{1\}$
proof
  from assms have $(i - 1) + 1 < length\ \mathcal{G}$ by simp
  hence simple: simple-group $(G|carrier := \mathcal{G} ! ((i - 1) + 1)) \ Mod \ \mathcal{G} ! (i - 1)$
  using simplefact by auto
  moreover from assms have $(i - 1) + 1 = i$ by auto
  ultimately have $G|carrier := \mathcal{G} ! ((i - 1) + 1)) \ Mod \ \mathcal{G} ! (i - 1) = G|carrier := \{1\} \ Mod \ \mathcal{G} ! (i - 1)$ using i by auto
  hence order $(G|carrier := \mathcal{G} ! ((i - 1) + 1)) \ Mod \ \mathcal{G} ! (i - 1)) = 1$ unfolding FactGroup-def order-def RCOSETS-def by force
  thus False using i simple unfolding simple-group-def simple-group-axioms-def by auto
qed

A composition series of a simple group always is its trivial one.

lemma (in composition-series) composition-series-simple-group:
  shows $(simple-group\ G) = (\mathcal{G} = \{[1]\},\ carrier\ G)$
proof
  assume $\mathcal{G} = \{[1]\},\ carrier\ G$
  with simplefact have simple-group $\ (G \ Mod \ \{1\})$ by auto
  moreover have the-elem $\in \ (G \ Mod \ \{1\}) \cong G$ by (rule trivial-factor-iso)
  ultimately show simple-group $G$ by (metis is-group simple-group.iso-simple)
next
  assume simple: simple-group $G$
  have length $\mathcal{G} > 1$
  proof (rule ccontr)
    assume $\neg 1 < length\ \mathcal{G}$
    hence length $\mathcal{G} = 1$ by (metis add.commute nat-less-cases not-add-less1 quotients-length)
    hence carrier $G = \{1\}$ using hd last by (metis composition-series-length-one composition-series-triv-group)
    hence order $G = 1$ unfolding order-def by auto
    with simple show False unfolding simple-group-def simple-group-axioms-def by auto
  qed
  moreover have length $\mathcal{G} \leq 2$
  proof (rule ccontr)

\[\text{def } k \equiv \text{length } G + 1 \quad \text{assume } \neg (\text{length } G \leq 2) \quad \text{hence gt2: length } G > 2 \text{ by simp} \]

\[\text{hence ksmall: } k + 1 < \text{length } G \text{ unfolding k-def by auto} \quad \text{from gt2 have carrier: } G ! (k + 1) = \text{carrier } G \text{ using notempty last last-conv-nth k-def} \quad \text{by (metis Nat.add-diff-associative Nat.diff-cancel \(\neg \) length } G \leq 2 \text{ add.commute nat-le-linear one-add-one)} \]

\[\text{from normal ksmall have } G ! k < G \quad \text{by simp} \quad \text{from simplefact ksmall have simplek: } \text{simple-group } (G[\text{carrier} := G ! (k + 1)]) \quad \text{Mod } G ! k \text{ by simp} \]

\[\text{from simplefact ksmall have simplek': } \text{simple-group } (G[\text{carrier} := G ! ((k - 1) + 1)]) \quad \text{Mod } G ! (k - 1) \text{ by auto} \quad \text{have } G ! k < G \text{ using carrier k-def gt2 normal ksmall by force} \]

\[\text{with simple have } (G ! k) = \text{carrier } G \lor (G ! k) = \{1\} \text{ unfolding simple-group-def simple-group-axioms-def by simp} \quad \text{thus False} \quad \text{proof (rule disjE)} \quad \text{assume } G ! k = \text{carrier } G \quad \text{hence } G[\text{carrier} := G ! (k + 1)] \quad \text{Mod } G ! k = G \text{ Mod (carrier } G) \text{ using carrier by auto} \]

\[\text{with simplek self-factor-not-simple show False by auto} \quad \text{next} \quad \text{assume } G ! k = \{1\} \quad \text{with ksmall k-def(gt2) show False using inner-elements-not-trivial by auto} \quad \text{qed} \quad \text{ultimately have length } G = 2 \text{ by simp} \]

\[\text{thus } G = \{\{1\}, \text{carrier } G\} \text{ by (rule length-two-unique)} \quad \text{qed} \]

Two consecutive elements in a composition series are distinct.

**Lemma (in composition-series) entries-distinct:**

- Assumes \(\text{finite:finite } (\text{carrier } G)\)
- Assumes \(i : i + 1 < \text{length } G\)
- Shows \(G ! i \neq G ! (i + 1)\)

**Proof**

\[\text{from finite have finite } (G ! (i + 1)) \quad \text{using i normal-series-subgroups subgroup-imp-subset rev-finite-subset by metis} \quad \text{hence fin:finite } (\text{carrier } (G[\text{carrier} := G ! (i + 1)])) \text{ by auto} \quad \text{from i have norm: } G ! i < (G[\text{carrier} := G ! (i + 1)]) \text{ by (rule normal)} \quad \text{assume } G ! i = G ! (i + 1) \quad \text{hence } G ! i = \text{carrier } (G[\text{carrier} := G ! (i + 1)]) \text{ by auto} \quad \text{hence } \text{carrier } (G[\text{carrier} := G ! (i + 1)]) \text{ Mod } (G ! i) = \{1 (G[\text{carrier} := G ! (i + 1)]) \text{ Mod } G ! i\} \quad \text{using norm fin normal, fact-group-trivial-iff by metis} \quad \text{hence } \neg \text{simple-group } ((G[\text{carrier} := G ! (i + 1)]) \text{ Mod } (G ! i)) \text{ by (metis simple-group.simple-not-trivial)} \quad \text{thus False by (metis i simplefact)} \quad \text{qed} \]
The normal series for groups of order \( p \ast q \) is even a composition series:

**lemma** (in group) \( \text{pq-order-composition-series} \):
- **assumes** \( \text{finite} : \text{finite} \) (carrier \( G \))
- **assumes** \( \text{order}G : \text{order} G = q \ast p \)
- **assumes** \( \text{prime}\_p : \text{prime} p \text{ and } \text{prime}\_q : \text{prime} q \text{ and } p \cdot q < q \)
- **shows** \( \text{composition-series} \ G \) \([\{1\}, \text{(THE} \ H \in \text{subgroups-of-size} \ q, \text{carrier} \ G)\] unfolding \( \text{composition-series-def} \) \( \text{composition-series-axioms-def} \)
- **apply** (auto)
- **using** \( \text{assms} \) **apply** (rule \( \text{pq-order-normal-series} \))
- **proof**
  - **def** \( H \equiv \text{THE} \ H \), \( H \in \text{subgroups-of-size} \ q \)
  - **from** \( \text{assms} \) **have** \( \text{exi} : \exists!Q. \ Q \in (\text{subgroups-of-size} \ q) \) by (auto simp; \( \text{pq-order-unique-subgrp} \))
  - **hence** \( \text{Hsize} : H \in \text{subgroups-of-size} \ q \) unfolding \( H\text{-def} \) by (metis \( \text{trivial-subgroup-is-normal} \) \( \text{is-group} \))
  - **hence** \( H_{\text{sub}G} : \text{subgroup} \ H \in \text{subgroups-of-size} \ q \) unfolding \( \text{subgroups-of-size-def} \) by auto
  - **then interpret** \( H_{\text{group}} : \text{group} \ G(\text{carrier} := H) \) by (metis \( \text{subgroup-imp-group} \))
  - **fix** \( i \)
    - **assume** \( i < \text{Suc} \ (\text{Suc} \ 0) \)
    - **hence** \( i = 0 \lor i = 1 \) by auto
    - **thus** \( \text{simple-group} \ (G(\text{carrier} := [H, \text{carrier} G] ! i) \text{ Mod} \ ([1], H, \text{carrier} G] ! i) \)
      - **proof**
        - **assume** \( i : i = 0 \)
          - **from** \( \text{Hsize} \) **have** \( \text{order}H : \text{order} (G(\text{carrier} := H)) = q \) unfolding \( \text{subgroups-of-size-def} \) \( \text{order-def} \) by simp
            - **hence** \( \text{order} (G(\text{carrier} := H) \text{ Mod} \ (1)) = q \) unfolding \( \text{FactGroup-def} \) using \( \text{card-rcosets-triv} \) \( \text{order-def} \)
              - by (metis \( \text{Hgroup.card-rcosets-triv} \) \( \text{HsubG} \) \( \text{finite} \) \( \text{monoid} \) \( \text{cases-scheme} \) \( \text{monoid.select-convs} \))
                - **partial-object, select-convs(1) partial-object, select-convs(1) subgroup-finite**
                    - **have** \( \text{normal} \ \{1\} (G(\text{carrier} := H)) \) by (metis \( \text{Hgroup.is-group} \) \( \text{Hgroup.normal-inv-iff} \) \( \text{HsubG} \) \( \text{group} \) \( \text{trivial-subgroup-is-normal} \) \( \text{is-group} \) \( \text{singleton-iff} \) \( \text{subgroup} \) \( \text{one-closed} \) \( \text{subgroup.subgroup-of-subgroup} \))
                        - **hence** \( \text{group} \ (G(\text{carrier} := H) \text{ Mod} \ (1)) \) by (metis \( \text{normal.factorgroup-is-group} \))
                            - **with** \( \text{order}H \) **primeq have** \( \text{simple-group} \ (G(\text{carrier} := H) \text{ Mod} \ (1)) \) by (metis \( \text{order} \) \( \text{G(\text{carrier} := H) \text{ Mod} \ (1)} \) = \( q \) \( \text{group} \) \( \text{prime-order-simple} \))
                                - **with** \( i \) **show** \( \text{thesis by simp} \)
    - **next**
      - **assume** \( i : i = 1 \)
        - **from** \( \text{assms exi have} \ H < \ G \) unfolding \( H\text{-def} \) by (metis \( \text{pq-order-subgrp-normal} \))
          - **hence** \( \text{group}GH : \text{group} \ (G \text{ Mod} H) \) by (metis \( \text{normal.factorgroup-is-group} \))
              - **from** \( \text{primeq have} \ q \neq 0 \) by (metis \( \text{zero-not-prime-nat} \))
                - **from** \( \text{HsubG finite orderG have} \ \text{card} \ \text{(rcosets} \ H) \ast \text{card} \ H = q \ast p \) unfolding \( \text{subgroups-of-size-def} \) using \( \text{lagrange} \) by simp
                    - **with** \( \text{Hsize have} \ \text{card} \ \text{(rcosets} \ H) \ast q = q \ast p \) unfolding \( \text{subgroups-of-size-def} \) by simp
                      - **with** \( q \neq 0 \) **have** \( \text{card} \ \text{(rcosets} \ H) = p \) by auto
                        - **hence** \( \text{order} \ (G \text{ Mod} H) = p \) unfolding \( \text{order-def} \) \( \text{FactGroup-def} \) by auto
                          - **with** \( \text{group}GH \) **primeq have** \( \text{simple-group} \ (G \text{ Mod} H) \) by (metis \( \text{group} \) \( \text{prime-order-simple} \))
                            - **with** \( i \) **show** \( \text{thesis by auto} \)
                              - **qed**
Prefixes of composition series are also composition series.

**Lemma** (in composition-series) composition-series-prefix-closed:
assumes $i \leq \text{length } G$ and $0 < i$
shows composition-series $(G[\text{carrier} := G \setminus (i - 1)])$ (take $i G$)
unfolding composition-series-def composition-series-axioms-def
proof auto
from assms show composition-series $(G[\text{carrier} := G \setminus (i - \text{Suc } 0)])$ (take $i G$) by
(metis One-nat-def normal-series-prefix-closed)
next
fix $j$
assume $j : \text{Suc } j < \text{length } G \text{ Suc } j < i$
with simplefact show simple-group $(G[\text{carrier} := G \setminus \text{Suc } j \text{ Mod } G \setminus j])$ by
(metis Suc-eq-plus1)
qed

The second element in a composition series is simple group.

**Lemma** (in composition-series) composition-series-snd-simple:
assumes $2 \leq \text{length } G$
shows simple-group $(G[\text{carrier} := G \setminus 1])$
proof –
from assms interpret compTake: composition-series $G[\text{carrier} := G \setminus 1]$ take 2 $G$ by
(metis add-diff-cancel-right' composition-series-prefix-closed one-add-one zero-less-numeral)
from assms have length (take 2 $G$) = 2 by (metis add-diff-cancel-right' append-take-drop-id
diff-diff-cancel length-append length-drop)
  hence (take 2 $G$) = $[\{1(G[\text{carrier} := G \setminus 1]\}, \text{carrier } (G[\text{carrier} := G \setminus 1])]$
by (rule compTake.length-two-unique)
thus ?thesis by (metis compTake.composition-series-simple-group)
qed

As a stronger way to state the previous lemma: An entry of a composition series is simple if and only if it is the second one.

**Lemma** (in composition-series) composition-snd-simple-iff:
assumes $i < \text{length } G$
shows $(\text{simple-group } (G[\text{carrier} := G \setminus i])) = (i = 1)$
proof
  assume simp:simple-group $(G[\text{carrier} := G \setminus i])$
  hence $G \setminus i \neq \{1\}$ using simple-group.simple-not-triv by force
  hence $i \neq 0$ using hd hd-cone-nth notempty by auto
  then interpret compTake: composition-series $G[\text{carrier} := G \setminus i]$ take $(\text{Suc } i)$
  $G$
  using assms composition-series-prefix-closed by (metis diff-Suc-1 less-eq-Suc-le zero-less-Suc)
  from simp have $(\text{take } (\text{Suc } i) G) = [(\{1G[\text{carrier} := G \setminus i]\}, \text{carrier } (G[\text{carrier} := G \setminus i])])$
  by (metis compTake.composition-series-simple-group)
hence \( \text{length } (\text{take} \ (\text{Suc} \ i) \ G) = 2 \) by auto
hence \( \text{min} \ (\text{length} \ G) \ (\text{Suc} \ i) = 2 \) by (metis length-take)
with assms have \( \text{Suc} \ i = 2 \) by force
thus \( i = 1 \) by simp

next
assume \( i:i = 1 \)
with assms have \( 2 \leq \text{length} \ G \) by simp
with \( i \) show \( \text{simple-group} \ (G\{\text{carrier} := G \setminus i\}) \) by (metis composition-series-snd-simple)
qed

The second to last entry of a normal series is not only a normal subgroup but actually even a maximal normal subgroup.

**lemma** (in composition-series) snd-to-last-max-normal:
assumes finite:finite \((\text{carrier} \ G)\)
assumes length:length \( \text{G} > 1 \)
shows max-normal-subgroup \( (G \setminus (\text{length} \ G - 2)) \ G \)
unfolding max-normal-subgroup-def max-normal-subgroup-axioms-def
proof (auto del: equalityI)
show \( G \) \((\text{length} \ G - 2) < G \) by (rule normal-series-snd-to-last)
next
def \( G' \equiv G \setminus (\text{length} \ G - 2) \)
from length have \( \text{length}(G) \cdot \text{length}G = \text{length}G - 1 \) by arith
from length have \( \text{length}G - 2 + 1 < \text{length}G \) by arith
with simplefact have \( \text{simple-group} \ (G\{\text{carrier} := G \setminus \text{length}G - 2 + 1\}) \)
Mod \( G' \) unfolding \( G'\)-def by auto
with \( \text{length}(G) \cdot \text{length}G = \text{length}G - 1 \) have \( \text{simple-last: simple-group} \ G \) using \( \text{less notempty} \)
last-conv-nth by fastforce
\{
  assume snd-to-last-eq: \( G' = \text{carrier} G \)
  hence \( \text{carrier} \ (G \ Mod G'') = \{1G \ Mod G'\} \)
  using normal-series-snd-to-last fact-group-trivial-iff unfolding \( G'\)-def by metis
  with snd-to-last-eq have \( \neg \text{simple-group} \ (G \ Mod G') \) by (metis self-factor-not-simple)
  with simple-last show False unfolding \( G'\)-def by auto
\}

have \( G'G:G' < G \) unfolding \( G'\)-def by (rule normal-series-snd-to-last)
fix \( J \)
assume \( J: J < G \) \( \neg G \) \( J \neq G \ G' \subseteq J \)
hence \( JG'GG':\text{rcosets}(G\{\text{carrier} := J\}) \) \( G' \subseteq G \ Mod G' \) using normality-factorization
normal-series-snd-to-last unfolding \( G'\)-def by auto
from \( G' G J(1,4) \) have \( G'J:G' < (G\{\text{carrier} := J\}) \) by (metis normal-imp-subgroup)
from \( \text{finite} \ J(1) \) have \( \text{fin}J: \text{finite} \ J \) by (auto simp: normal-imp-subgroup subgroup-finite)
from \( JG'GG' \) simple-last have \( \text{rcosets}(G\{\text{carrier} := J\}) \) \( G' = \{1G \ Mod G'\} \lor \text{rcosets}(G\{\text{carrier} := J\}) \) \( G' = \text{carrier} \ (G \ Mod G') \)
unfolding simple-group-def simple-group-axioms-def by auto
thus \( False \)
proof
 assume rcosets \( G \| \text{carrier} := J \| G' = \{1 \ Mod \ G'\} \)
hence rcosets \( G \| \text{carrier} := J \| G' = \{1(G\| \text{carrier} := J) \ Mod \ G'\} \) unfolding FactGroup-def by simp
hence \( G' = J \) using \( G'J \) finJ normal.fact-group-trivial-iff unfolding FactGroup-def by fastforce
hence \( G' = \begin{cases} J & \text{if facts-eq:rcosets} \ G \| \text{carrier} := J \| G' = \text{carrier} \ (G \ Mod \ G') \end{cases} \) unfolding FactGroup-def by simp
hence \( G' = J \) using \( G'J \) finJ normal.normal-imp-subgroup subgroup-imp-subset
with \( J(2) \) show False by simp
next
assume facts-eq:rcosets \( G \| \text{carrier} := J \| G' = \text{carrier} \ (G \ Mod \ G') \)
have \( J = \text{carrier} \ G \) proof
show \( J \subseteq \text{carrier} \ G \) using \( J(1) \) normal-imp-subgroup subgroup-imp-subset
proof
fix \( x \)
assume \( x : x \in \text{carrier} \ G \)
hence \( G' \#> x \in \text{carrier} \ (G \ Mod \ G') \) unfolding FactGroup-def RCOSETS-def by auto
hence \( G' \#> x \in \text{rcosets} \ G \| \text{carrier} := J \| G' = \text{carrier} \ (G \ Mod \ G') \) using facts-eq by auto
then obtain \( j \) where \( j : j \in J \ G' \#> x = G' \#> j \) unfolding RCOSETS-def by force
hence \( x \in G' \#> j \) using \( G'G \) normal-imp-subgroup \( x \) repr-independenceD by fastforce
then obtain \( g' \) where \( g' : g' \in G' x = g' \otimes j \) unfolding r-coset-def by auto
hence \( g' \in J \) using \( G'J \) normal-imp-subgroup subgroup-imp-subset by force
with \( g'(2) j(1) \) show \( x \in J \) using \( J(1) \) normal-imp-subgroup subgroup.m-closed by fastforce
qed
qed

For the next lemma we need a few facts about removing adjacent duplicates.

lemma remdups-adj-obtain-adjacency:
assumes \( i + 1 < \text{length} \ (\text{remdups-adj} \ xs) \) \( \text{length} \ xs > 0 \)
obtains \( j \) where \( j + 1 < \text{length} \ xs \)
(remdups-adj xs) ! i = xs ! j (remdups-adj xs) ! (i + 1) = xs ! (j + 1)
using \( \text{assms} \) proof (induction \( xs \) arbitrary; \( i \) \( \text{thesis} \))
case Nil
hence \( \text{False} \) by (metis length-greater-0-conv)
thus \( \text{thesis} .. \)
next
case \( \text{Cons} \ x \ xs \)
hence \( xs \neq [] \) using Divides.div_less Suc_eq1 Zero_not_Suc div_eq_dividend_iff
list.size(3,4) plusNat.add_0 remdups-adj.simps(2) by metis
then obtain \( y \) \( xs' \) where \( xs;xs = y \neq xs' \) by (metis list.exhaust)
from \( xs \neq [] \) have lenxs:length xs > 0 by simp
from \( xs \) have rem:remdups-adj \( (x \neq xs) = (if x = y \text{ then remdups-adj} (y \neq xs') \text{ else } x \neq \text{ remdups-adj} (y \neq xs')) \) using remdups-adj.simps(3) by auto
show thesis
proof (cases \( x = y \))
  case True
  with \( \text{ rem xs have rem2:remdups-adj} (x \neq xs) = \text{ remdups-adj} \) \( xs \) by auto
  with Cons(3) have \( i + 1 < \text{ length (remdups-adj} \) \( xs) \) by simp
  with Cons.HH lenxs obtain \( k \) where \( j \cdot k + 1 < \text{ length xs remdups-adj} \) \( xs \) ! \( i = xs \) ! \( k \) 
    remdups-adj \( xs \) ! \( (i + 1) = xs \) ! \( (k + 1) \) by auto
    thus thesis using Cons(2) rem2 by auto
next
  case False
  with \( \text{ rem xs have rem2:remdups-adj} (x \neq xs) = x \# \text{ remdups-adj} \) \( xs \) by auto
  show thesis
  proof (cases \( i \))
    case 0
    have \( 0 + 1 < \text{ length (x \# xs) using lenxs by auto} \)
    moreover have \( \text{ remdups-adj} (x \# xs) ! i = (x \# xs) ! 0 \) using \( \text{ xs rem2 0 by simp} \)
    also have \( \ldots = x \) by simp
    also have \( \ldots = (x \# xs) ! 0 \) by simp
    finally show \( ? \text{thesis.} \)
    qed
    moreover have \( \text{ remdups-adj} (x \# xs) ! (i + 1) = (x \# xs) ! (0 + 1) \)
    proof --
      have \( \text{ remdups-adj} (x \# xs) ! (i + 1) = (x \# \text{ remdups-adj} (y \# xs')) ! 1 \)
      using \( \text{ xs rem2 0 by simp} \)
      also have \( \ldots = \text{ remdups-adj} (y \# xs') ! 0 \) by simp
      also have \( \ldots = (y \# (\text{ remdups (y \# xs'))} ! 0 \) by (metis nthCons't remdups-adjConsAlt)
      also have \( \ldots = y \) by simp
      also have \( \ldots = (x \# xs) ! (0 + 1) \) unfolding \( \text{ xs by simp} \)
      finally show \( ? \text{thesis.} \)
      qed
    ultimately show thesis by (rule Cons.prems(1))
next
  case (Suc \( k \))
  with Cons(3) have \( k + 1 < \text{ length (remdups-adj} (x \# xs)) - 1 \) by auto
  also have \( \ldots \leq \text{ length (remdups-adj} \) \( xs) + 1 - 1 \) by (metis One_nat_def le_refl list.size(4) rem2)
  also have \( \ldots = \text{ length (remdups-adj} \) \( xs) \) by simp
  finally have \( k + 1 < \text{ length (remdups-adj} \) \( xs \)).
with Cons.IH lenxs obtain j where \( j : j + 1 < \text{length} \; xs \) remdups-adj \( xs \) ! \( k = xs ! j \)

remdups-adj \( xs ! (k + 1) = xs ! (j + 1) \) by auto

from \( j(1) \) have \( Suc \; j + 1 < \text{length} \; (x \# xs) \) by simp

moreover have remdups-adj \( (x \# xs) ! i = (x \# xs) ! (Suc \; j) \)

proof –

have remdups-adj \( (x \# xs) ! i = (x \# \text{remdups-adj} \; xs) ! (Suc \; j) \)

using rem2 by simp

also have \( \ldots = \text{remdups-adj} \; xs ! k \) using Suc by simp

also have \( \ldots = xs ! j \) using \( j(2) \).

also have \( \ldots = (x \# xs) ! (Suc \; j) \) by simp

finally show ?thesis.

qed

moreover have remdups-adj \( (x \# xs) ! (i + 1) = (x \# \text{remdups-adj} \; xs) ! (i + 1) \)

proof –

have remdups-adj \( (x \# xs) ! (i + 1) = (x \# \text{remdups-adj} \; xs) ! (i + 1) \)

using rem2 by simp

also have \( \ldots = \text{remdups-adj} \; xs ! (k + 1) \) using Suc by simp

also have \( \ldots = xs ! (j + 1) \) using \( j(3) \).

also have \( \ldots = (x \# xs) ! (Suc \; j + 1) \) by simp

finally show ?thesis.

qed

ultimately show thesis by (rule Cons.prems(1))

qed

lemma hd-remdups-adj[simp]: \( \text{hd} \; \text{remdups-adj} \; xs = \text{hd} \; xs \)

by (induction \( xs \) rule: remdups-adj.induct) simp-all

lemma remdups-adj-adjacent:

\( Suc \; i < \text{length} \; \text{remdups-adj} \; xs \) \( \Rightarrow \) remdups-adj \( xs ! i \neq \text{remdups-adj} \; xs ! Suc \; i \)

proof (induction \( xs \) arbitrary: \( i \) rule: remdups-adj.induct)

case \( 3 \; x \; y \; xs \; i \)

thus ?case by (cases \( i \), cases \( x = y \) ) (simp, auto simp: hd-conv-nth[symmetric])

qed simp-all

Intersecting each entry of a composition series with a normal subgroup of \( G \) and removing all adjacent duplicates yields another composition series.

lemma (in composition-series) intersect-normal:

assumes finite:finite (carrier \( G \))

assumes \( KG : K \triangleleft G \)

shows composition-series \( (G \langle \text{carrier} := K \rangle) \; \text{remdups-adj} \; (\text{map} \; (\lambda H. \; K \cap H) \; G)) \)


apply (auto simp only: conjI del: equalityI)

proof –

show group \( (G \langle \text{carrier} := K \rangle) \) using KG normal-imp-subgroup subgroup-imp-group
by auto

next
— Show, that removing adjacent duplicates doesn’t result in an empty list.
assume remdups-adj \( (\map (\op \cap K) \mathcal{G}) = [] \)
hence \( \map (\op \cap K) \mathcal{G} = [] \) by (metis remdups-adj.Nil-iff)
hence \( \mathcal{G} = [] \) by (metis Nil-is-map-conv)
with notempty show False..

next
— Show, that the head of the reduced list is still the trivial group
have \( \mathcal{G} = \{1\} \neq \tl \mathcal{G} \) using notempty hd by (metis list.sel(1,3) neq-Nil-conv)
hence \( \map (\op \cap K) \mathcal{G} = \map (\op \cap K) (\{1\} \neq \tl \mathcal{G}) \) by simp
hence remdups-adj \( (\map (\op \cap K) \mathcal{G}) = \map \map (\op \cap K) ((\cap \cap \{1\}) \neq (\cap \cap (\op \cap K) (\tl \mathcal{G}))) \) by simp
also have \( \dots = (\cap \cap \{1\}) \neq \tl \map \map (\op \cap K) ((\cap \cap (\op \cap K) (\tl \mathcal{G}))) \) by simp
finally have \( \tl \map \map \map (\op \cap K) (\tl \mathcal{G}) = \cap K \) using list.sel(1)
by metis
thus \( \tl \map \map \map (\op \cap K) (\tl \mathcal{G}) = \{1\}_{\tl \mathcal{G}} \)
using KG normal-imp-subgroup subgroup.one-closed by force

next
— Show that the last entry is really \( K \cap G \). Since we don’t have a lemma ready
to talk about the last entry of a reduced list, we reverse the list twice.

have \( \rev \mathcal{G} = (\cap \tl G) \neq \tl (\rev \mathcal{G}) \) by (metis list.sel(1,3) last last-rev
neq-Nil-conv notempty \( \rev \map (\op \cap K) \mathcal{G} = \map (\op \cap K) \) \( (\cap \tl G) \neq (\cap (\op \cap K) (\tl G)) \) by (metis rev-map)

hence \( \rev \map \map \rev (\map (\op \cap K) \mathcal{G}) = (\cap G (\cap (\op \cap G)) \neq (\map (\op \cap G) (\cap (\tl \mathcal{G}))) \) by simp

have last \( \map \map \map \map (\op \cap K) (\tl G) = \tl \map \map \map \map (\op \cap K) (\tl (\tl \mathcal{G})) \)
by (metis hd-rev map-is-Nil-conv notempty remdups-adj-nil-iff)
also have \( \dots = \tl \map \map \map \map (\op \cap K) (\tl (\tl \mathcal{G})) \) by (metis remdups-adj-rev)
also have \( \dots = \tl \map \map \map \map (\tl (\tl \mathcal{G})) \) \( (\cap \tl G (\cap (\op \cap G)) \neq (\map \map (\op \cap G) (\cap (\tl \mathcal{G})))) \) by (metis list.sel(1)
remdups-adj-Cons-alt)
also have \( \dots = K \) using KG normal-imp-subgroup subgroup.imp-subset by force
finally show last \( \map \map \map \map (\op \cap K) (\tl \mathcal{G}) = \tl \tl \mathcal{G} \) (\tl G (\cap \tl G))
by auto

next
— The induction step, using the second isomorphism theorem for groups.
fix \( j \)
assume \( j ; j + 1 \leq \len \map (\op \cap K) \mathcal{G} \)
have \( KG \) notempty :\( (\map (\op \cap K) \mathcal{G}) \) \( = [] \) using notempty by (metis Nil-is-map-conv)
with \( j \) obtain \( i \) where \( i ; i + 1 \leq \len \map (\op \cap K) \mathcal{G} \)
(remdups-adj \( (\map (\op \cap K) \mathcal{G})) ! j = (\map (\op \cap K) \mathcal{G}) ! i \)
(remdups-adj \( (\map (\op \cap K) \mathcal{G})) ! (j + 1) = (\map (\op \cap K) \mathcal{G}) ! (i + 1) \)
using remdups-adj-obtain-adjacency by force
from \( i (1) \) have \( i ; i + 1 \leq \len \mathcal{G} \) by (metis length-map)
hence \( G\!iSi ; ! i \triangleleft G\{\text{carrier} := \emptyset ! (i + 1)\} \) by (metis normal)

hence \( G\!iSi' ; ! i \subseteq \emptyset ! (i + 1) \) using normal-imp-subgroup subgroup-imp-subset by force

from \( i' \) have \( \text{finG}!Si ; \text{finite} (\emptyset ! (i + 1)) \) using normal-series-subgroups finite by (metis subgroup-finite)

from \( G\!iSi KG \) \( i' \) normal-series-subgroups have \( G\!iSiapulte K ; (\emptyset ! (i + 1)) \)

using second-isomorphism-grp.normal-subgrp-intersection-normal

unfolding second-isomorphism-grp-def second-isomorphism-grp-axioms-def by auto

with \( G\!iSi \) have \( \emptyset ! i \cap (\emptyset ! (i + 1) \triangleleft K) \triangleleft G\{\text{carrier} := \emptyset ! (i + 1)\} \)

by (metis group.normal-subgroup-intersect group subgroup-imp-group i' is-group normal-series normal-series-normal-series-subgroups)

hence \( K \cap (\emptyset ! i \cap \emptyset ! (i + 1)) \triangleleft G\{\text{carrier} := \emptyset ! (i + 1)\} \) by (metis inf-commute inf-left-commute)

hence \( KG\!iSi ; G\!iSi KG i \)

from \( \text{second-isomorphism-grp-def second-isomorphism-grp-axioms-def} \) second-isomorphism-grp

ultimately have \( \text{fstgoal} K \cap \emptyset ! i \triangleleft G\{\text{carrier} := \emptyset ! (i + 1)\}, \text{carrier} := K \cap \emptyset ! (i + 1) \)

using group.normal-restrict-supergroup by force

thus \( \text{rendups-adj} \ (\text{map} \ ((\text{op} \cap K) \emptyset)) \ j \triangleleft G\{\text{carrier} := K, \text{carrier} := \text{rendups-adj} \ (\text{map} \ ((\text{op} \cap K) \emptyset)) \) \)

using \( i \) by auto

from \( \text{simplefact} \) have \( G\!iSi ; \text{simple-group} (G\{\text{carrier} := \emptyset ! (i + 1)\}) \text{ Mod} \emptyset ! \) i using \( i' \) by simp

hence \( G\!iSi ; \text{max-normal-subgroup} (\emptyset ! i) \ (G\{\text{carrier} := \emptyset ! (i + 1)\}) \)

using normal.max-normal-simple-quotient G\!iSi finG\!Si by force

from \( G\!iSiapulte K ; i \triangleleft G\{\text{carrier} := \emptyset ! (i + 1)\} \emptyset ! (i + 1) \)

\( \cap K \triangleleft G\{\text{carrier} := \emptyset ! (i + 1)\} \)

using group normal-subgroup-set-mult-closed by simp

hence \( \emptyset ! i \triangleleft G\{\text{carrier} := \emptyset ! (i + 1)\} \)

unfolding set-mult-def by auto

hence \( \emptyset ! i \triangleleft G\{\text{carrier} := \emptyset ! (i + 1)\} \)

using inf-commute by metis

moreover have \( \emptyset ! i \subseteq \emptyset ! i \triangleleft G\{\text{carrier} := \emptyset ! (i + 1)\} \)


unfolding second-isomorphism-grp-def second-isomorphism-grp-axioms-def

using subKGS\!iSi G\!iSi normal-imp-subgroup by fastforce

hence \( \emptyset ! i \subseteq \emptyset ! i \triangleleft G\{\text{carrier} := \emptyset ! (i + 1)\} \)

unfolding set-mult-def by auto

ultimately have \( KG\!iSi ; i \triangleleft K \cap \emptyset ! (i + 1) = \emptyset ! i \cap \emptyset ! i \triangleleft K \cap \emptyset ! (i + 1) \)

using G\!iSi unfolding max-normal-subgroup-def max-normal-subgroup-axioms-def by auto

obtain \( \phi \) where \( \phi \in (G\{\text{carrier} := K \cap \emptyset ! (i + 1)\}) \text{ Mod} (\emptyset ! i \cap (K \cap \emptyset !) \text{ normal-imp-subset by force}


\[(i + 1)\]
\[\vDash (G[carrier := \emptyset ! i <\#> G[carrier := \emptyset ! (i + 1)]) \text{ Mod } \emptyset ! i] \]

using second-isomorphism-grp.normal-intersection-quotient-isom

unfolding second-isomorphism-grp-def second-isomorphism-grp-axioms-def

using \(GSi \subset KGSiGSi \text{ normal-imp-subgroup by fastforce}

hence \(\varphi \in (G[carrier := K \cap \emptyset ! (i + 1)]) \text{ Mod } (K \cap \emptyset ! (i + 1) \cap \emptyset ! i)) \]
\[\vDash (G[carrier := \emptyset ! i <\#> G[carrier := \emptyset ! (i + 1)]) \text{ Mod } \emptyset ! i] \]

by (metis inf-commute)

hence \(\varphi \in (G[carrier := K \cap \emptyset ! (i + 1)]) \text{ Mod } (K \cap (\emptyset ! (i + 1) \cap \emptyset ! i))) \]
\[\vDash (G[carrier := \emptyset ! i <\#> G[carrier := \emptyset ! (i + 1)]) \text{ Mod } \emptyset ! i] \]

by (metis Int-assoc)

hence \(\varphi \in (G[carrier := K \cap \emptyset ! (i + 1)]) \text{ Mod } (K \cap \emptyset ! i)) \]
\[\vDash (G[carrier := \emptyset ! i <\#> G[carrier := \emptyset ! (i + 1)]) \text{ Mod } \emptyset ! i] \]

by (metis GiSi Int-absorb2 Int-commute)

hence \(\varphi \in (G[carrier := K \cap \emptyset ! (i + 1)]) \text{ Mod } (K \cap \emptyset ! i)) \]
\[\vDash (G[carrier := \emptyset ! i <\#> K \cap \emptyset ! (i + 1)]) \text{ Mod } \emptyset ! i] \]

unfolding set-mult-def by auto

from fstgoal have \(KGsiKiGigroup\text{-}group (G[carrier := K \cap \emptyset ! (i + 1)]) \text{ Mod } (K \cap \emptyset ! i)) \text{ using normal-factorgroup-is-group by auto}

from \(KGdisj\text{-}show\) simple-group \((G[carrier := K, carrier := \text{ remdups-adj (map (op \cap K) \emptyset ! (j + 1))) \text{ Mod } \text{ remdups-adj (map (op \cap K) \emptyset ! j))}} \)

proof auto

have \(\text{ groupGi\text{-}group (G[carrier := \emptyset ! i]) \text{ using } i' \text{ normal-series-subgroups subgroup-imp\text{-}group by auto}

assume \(\emptyset ! i <\#> K \cap \emptyset ! \text{ Suc } i = \emptyset ! i \)

with \(\varphi \text{ have } \varphi \in (G[carrier := K \cap \emptyset ! (i + 1)]) \text{ Mod } (K \cap \emptyset ! i)) \)
\[\vDash (G[carrier := \emptyset ! i] \text{ Mod } \emptyset ! i] \]

by auto

moreover obtain \(\psi\) where \(\psi \in (G[carrier := \emptyset ! i] \text{ Mod } (carrier (G[carrier := \emptyset ! i]))) \vDash (G[carrier := \{1G[carrier := \emptyset ! i])]) \)

using group.self-factor-iso groupGi by force

ultimately obtain \(\pi\) where \(\pi \in (G[carrier := K \cap \emptyset ! (i + 1)]) \text{ Mod } (K \cap \emptyset ! i)) \)
\[\vDash (G[carrier := \{1G[carrier := \emptyset ! i])]) \]

using KGsiKiGigroup group.iso-trans by fastforce

hence order \((G[carrier := K \cap \emptyset ! (i + 1)]) \text{ Mod } (K \cap \emptyset ! i)) = \text{order}
\[\vDash (G[carrier := \{1G[carrier := \emptyset ! i])]) \text{ by (metis iso-order-closed)} \]

hence order \((G[carrier := K \cap \emptyset ! (i + 1)]) \text{ Mod } (K \cap \emptyset ! i)) = 1 \text{ unfolding order-def by auto}

hence carrier \((G[carrier := K \cap \emptyset ! (i + 1)]) \text{ Mod } (K \cap \emptyset ! i)) = \{1G[carrier := K \cap \emptyset ! (i + 1)]) \text{ Mod } (K \cap \emptyset ! i)\}

using group.order-one-triv-iff KGsiKiGigroup by auto

moreover from fstgoal have \(K \cap \emptyset! i \in G[carrier := K \cap \emptyset ! (i + 1)]) \text{ by auto}

moreover from finGSi have finite \((carrier (G[carrier := K \cap \emptyset ! (i + 1)])]) \text{ by auto}

ultimately have \(K \cap \emptyset ! i = \text{ carrier } (G[carrier := K \cap \emptyset ! (i + 1)]) \)

by (metis normal факт\text{-}group\text{-}trivial\text{-}iff)

hence \((\text{ remdups-adj (map (op \cap K) \emptyset !))} \text{ j = (remdups-adj (map (op \cap K) \emptyset !))} \text{ by auto})
\( \emptyset \}) \leadsto (j + 1) \text{ using } i \text{ by } auto

with \( j \) have False using remdups-adj-adjacent KGNempty Suc-eq-plus1 by metis

thus simple-group (G(carrier := remdups-adj (map (op ∩ K) \( \emptyset \}) ! Suc j)) Mod remdups-adj (map (op ∩ K) \( \emptyset \}) ! j).

next

assume \( \emptyset \} ! i < \# > K \cap \emptyset ! Suc i = \emptyset ! Suc i 

moreover with \( \varphi \) have \( \varphi \in (G(carrier := K \cap \emptyset ! (i + 1)) Mod (K \cap \emptyset ! i)) \equiv (G(carrier := \emptyset ! (i + 1)) Mod \emptyset ! i)) \text{ by auto }

then obtain \( \varphi' \) where \( \varphi' \in (G(carrier := \emptyset ! (i + 1)) Mod \emptyset ! i) \equiv (G(carrier := K \cap \emptyset ! (i + 1)) Mod (K \cap \emptyset ! i)) 

using KGsiKGi-group iso-sym by auto

with Gsimple KGsiKGi-group have simple-group (G(carrier := K \cap \emptyset ! (i + 1)) Mod (K \cap \emptyset ! i)) by (metis simple-group.iso-simple)

with i show simple-group (G(carrier := remdups-adj (map (op ∩ K) \( \emptyset \}) ! Suc j)) Mod remdups-adj (map (op ∩ K) \( \emptyset \}) ! j) by auto

qed

lemma (in group) composition-series-extend:

assumes composition-series (G(carrier := H)) \( \emptyset \)

assumes simple-group (G Mod H) H ∩ G

shows composition-series G (\( \emptyset \} @ [carrier G])

unfolding composition-series-def composition-series-axioms-def

proof auto

from assms(1) interpret comp\( \emptyset \} : composition-series G(carrier := H) \( \emptyset \).

show normal-series G (\( \emptyset \} @ [carrier G]) using assms(3) comp\( \emptyset \}, is-normal-series

by (metis normal-series-extend)

fix i

assume i:i < length \( \emptyset \)

show simple-group (G(carrier := (\( \emptyset \} @ [carrier G]) ! Suc i)) Mod (\( \emptyset \} @ [carrier G]) ! i)

proof (cases i = length \( \emptyset \} = 1)

case True

hence (\( \emptyset \} @ [carrier G]) ! Suc i = carrier G by (metis i diff-Suc-1 lessE nth-append-length)

moreover have (\( \emptyset \} @ [carrier G]) ! i = \( \emptyset \} ! i) by (metis butlast-snoc i nth-butlast)

hence (\( \emptyset \} @ [carrier G]) ! i = H using True last-conv-nth comp\( \emptyset \}, notempty comp\( \emptyset \}, last by auto

ultimately show ?thesis using assms(2) by auto

next

case False

hence Suc i < length \( \emptyset \} using i by auto

hence (\( \emptyset \} @ [carrier G]) ! Suc i = \( \emptyset \} ! Suc i using nth-append by metis

moreover from i have (\( \emptyset \} @ [carrier G]) ! i = \( \emptyset \} ! i using nth-append by metis

ultimately show ?thesis using (Suc i < length \( \emptyset \}) comp\( \emptyset \}, simplefact by auto

qed

qed
lemma (in composition-series) entries-mono:
  assumes i ≤ j j < length G
  shows G ! i ⊆ G ! j
using assms proof (induction j - i arbitrary: i j)
  case \emptyset
  hence i = j by auto
  thus G ! i ⊆ G ! j by auto
next
  case (Suc k i j)
  hence i':i + (Suc k) = j i + 1 < length G by auto
  hence i'j:i + 1 ≤ j by auto
  have G ! i ⊆ G ! (i + 1) using i' normal normal-imp-subgroup subgroup-imp-subset
  by force
  moreover have j - (i + 1) = k j < length G using Suc assms by auto
  hence G ! (i + 1) ⊆ G ! j using Suc(1) i j by auto
  ultimately show G ! i ⊆ G ! j by simp
qed

end

theory GroupIsoClasses
imports
  Groups
  List
  Coset
begin

9 Isomorphism Classes of Groups

We construct a quotient type for isomorphism classes of groups.

typedef 'a group = {G :: 'a monoid. group G}
proof
  show \\(\forall a. \{\text{carrier} = \{a\}, \text{mult} = (\lambda x y. x), \text{one} = a\} \in \{G. \text{group G}\}\)
  unfolding group-def group-axioms-def monoid-def Units-def by auto
qed
definition group-iso-rel :: 'a group ⇒ 'a group ⇒ bool
  where group-iso-rel G H = (\exists \varphi. \varphi \in \text{Rep-group G} \cong \text{Rep-group H})

quotient-type 'a group-iso-class = 'a group / group-iso-rel
morphisms Rep-group-iso Abs-group-iso
proof (rule equivpI)
  show reflp group-iso-rel
proof (rule reflpI)
  fix G :: 'b group
  show group-iso-rel G G
unfolding group-iso-rel-def using Rep-group iso-refl by auto
qed
next
show symp group-iso-rel
proof (rule sympI)
  fix G H :: 'b group
  assume group-iso-rel G H
  then obtain ϕ where ϕ ∈ Rep-group G ≃ Rep-group H unfolding group-iso-rel-def by auto
  then obtain ϕ' where ϕ' ∈ Rep-group H ≃ Rep-group G using group.iso-sym
  Rep-group by fastforce
  thus group-iso-rel H G unfolding group-iso-rel-def by auto
qed
next
show transp group-iso-rel
proof (rule transpI)
  fix G H I :: 'b group
  assume group-iso-rel G H group-iso-rel H I
  then obtain ϕ ψ where ϕ ∈ Rep-group G ≃ Rep-group H ψ ∈ Rep-group H
  ≃ Rep-group I unfolding group-iso-rel-def by auto
  then obtain π where π ∈ Rep-group G ≃ Rep-group I using group.iso-trans
  Rep-group by fastforce
  thus group-iso-rel G I unfolding group-iso-rel-def by auto
qed
qed

This assigns to a given group the group isomorphism class

definition (in group) iso-class :: 'a group-iso-class
  where iso-class = Abs-group-iso (Abs-group (monoid.truncate G))

Two isomorphic groups do indeed have the same isomorphism class:

lemma iso-classes-iff:
  assumes group G
  assumes group H
  shows (∃ϕ. ϕ ∈ G ≃ H) = (group.iso-class G = group.iso-class H)
proof –
  from assms(1,2) have groups:group (monoid.truncate G) group (monoid.truncate H)
  unfolding monoid.truncate-def group-def group-axioms-def Units-def monoid-def by auto
  have (∃ϕ. ϕ ∈ G ≃ H) = (∃ϕ. ϕ ∈ (monoid.truncate G) ≃ (monoid.truncate H))
  unfolding iso-def hom-def monoid.truncate-def by auto
  also have ... = group-iso-rel (Abs-group (monoid.truncate G)) (Abs-group (monoid.truncate H))
  unfolding group-iso-rel-def using groups group.Abs-group-inverse by (metis mem-Collect-eq)
  also have ... = (group.iso-class G = group.iso-class H) using group.iso-class-def
  assms group-iso-class.abs-eq-iff by metis
Finally show \( \text{thesis} \).
qed
end

theory JordanHolder
imports
CompositionSeries
MaximalNormalSubgroups
Multiset
GroupIsoClasses
begin

10 The Jordan-Hölder Theorem

locale jordan-hoelder =
group + comp\(\Sigma\) : composition-series \( G \) \( \mathcal{H} \)
+ comp\(\Gamma\) : composition-series \( G \) \( \mathcal{G} \) for \( \mathcal{H} \) and \( \mathcal{G} \)
+ assumes finite : finite (carrier \( G \))

Before we finally start the actual proof of the theorem, one last lemma:
Cancelling the last entry of a normal series results in a normal series with
quotients being all but the last of the original ones.

lemma (in normal-series) quotients-butlast:
  assumes length \( \mathcal{G} \) \( > 1 \)
  shows butlast quotients = normal-series.quotients (\( G[\text{carrier} := \mathcal{G} ! (\text{length} \mathcal{G} - 1 - 1)]\)) (\( \text{take} (\text{length} \mathcal{G} - 1) \mathcal{G} \))
proof (rule nth-equalityI )
  def \( n \equiv \text{length} \mathcal{G} - 1 \)
  hence \( n = \text{length} (\text{take} n \mathcal{G}) \) \( n > 0 \) \( n < \text{length} \mathcal{G} \) using assms notempty by auto
  interpret normal\(\mathcal{G}^{\text{butlast}}\) : normal-series (\( G[\text{carrier} := \mathcal{G} ! (n - 1)]\)) \( \text{take} n \mathcal{G} \)
  using normal-series-prefix-closed \( \langle n > 0 \rangle \) \( \langle n < \text{length} \mathcal{G} \rangle \) by auto
  have length (butlast quotients) = length quotients - 1 by (metis length-butlast)
  also have \( \ldots = \text{length} \mathcal{G} - 1 - 1 \) by (metis add-diff-cancel-right' quotients-length)
  also have \( \ldots = \text{length} (\text{take} n \mathcal{G}) - 1 \) by (metis (n = length (take n \mathcal{G})). n-def)
  also have \( \ldots = \text{length normal}\mathcal{G}^{\text{butlast}}.quotients \) by (metis normal\(\mathcal{G}^{\text{butlast}}\).quotients-length
  diff-add-inverse2)
  finally show length (butlast quotients) = length normal\(\mathcal{G}^{\text{butlast}}\).quotients .
  have \( \forall i < \text{length} (\text{butlast quotients}). \text{butlast quotients} ! i = \text{normal}\mathcal{G}^{\text{butlast}}.quotients ! i \)
  proof auto
    fix \( i \)
    assume \( i : i < \text{length} \text{quotients} - \text{Suc} 0 \)
    hence \( i' : i < \text{length} \mathcal{G} - 1 < n i + 1 < n \) unfolding n-def using quotients-length by auto

48
from \( i \) have \( \text{butlast quotients}! \; \text{i} = \text{quotients}! \; \text{i} \) by (metis One-nat-def length-butlast nth-butlast)
also have \( \ldots = G[[\text{carrier} := \mathcal{G} ! (i + 1)]], ! \) \( \text{Mod} \; \mathcal{G} ! \; \text{i} \) unfolding quotients-def
using \( i'(1) \) by auto
also have \( \ldots = G[[\text{carrier} := (\text{take n} \; \mathcal{G}) ! (i + 1)]], ! \) \( \text{Mod} \; (\text{take n} \; \mathcal{G}) ! \; \text{i} \) using
\( i'(2,3) \) nth-take by metis
also have \( \ldots = \text{normal}!\mathcal{G} \text{butlast.quotients}! \; \text{i} \) unfolding normal\mathcal{G} butlast.quotients-def
using \( i' \) by fastforce
finally show \( \text{butlast} \; \text{normal-series.quotients} G \; \mathcal{G} ! \; \text{i} \) = \( \text{normal-series.quotients} (G[[\text{carrier} := \mathcal{G} ! (\text{length} \; \mathcal{G} - 1 - 1)]]) \) (take (length \; \mathcal{G} - 1) \; \mathcal{G}) ! \; \text{i}
unfolding n-def by auto
qed

The main part of the Jordan–Hölder theorem is its statement about the uniqueness of a composition series. Here, uniqueness up to reordering and isomorphism is modelled by stating that the multisets of isomorphism classes of all quotients are equal.

**Theorem jordan-hoelder-multisets:**
assumes group \( G \)
assumes finite (carrier \( G \))
assumes composition-series \( G \; \mathcal{G} \)
assumes composition-series \( G \; \mathcal{H} \)
shows mset (map group.iso-class (normal-series.quotients \( G \; \mathcal{G} \)))
= mset (map group.iso-class (normal-series.quotients \( G \; \mathcal{H} \)))
using assms
proof (induction length \( \mathcal{G} \) arbitrary: \( \mathcal{G} \; \mathcal{H} \) G rule: fall-nat-induct)
case (1 \( \mathcal{G} \; \mathcal{H} \) \( G \))
then interpret comp\( \mathcal{G} \): composition-series \( G \; \mathcal{G} \) by simp
from 1 interpret comp\( \mathcal{H} \): composition-series \( G \; \mathcal{H} \) by simp
from 1 interpret grp\( \mathcal{G} \): group \( G \) by simp
show ?case
proof (cases length \( \mathcal{G} \) \( \leq \) 2)
next
case True
hence length \( \mathcal{G} = 0 \lor \text{length} \; \mathcal{G} = 1 \lor \text{length} \; \mathcal{G} = 2 \) by arith
with comp\( \mathcal{G} \).notempty have length \( \mathcal{G} = 1 \lor \text{length} \; \mathcal{G} = 2 \) by simp
thus ?thesis
proof auto
— First trivial case: \( \mathcal{G} \) is the trivial group.
  assume length \( \mathcal{G} = \text{Suc} \; 0 \)
  hence length:length \( \mathcal{G} = 1 \) by simp
  hence length \( \| + 1 = \text{length} \; \mathcal{G} \) by auto
moreover from length have char\( \mathcal{G} : \mathcal{G} = [1_G] \) by (metis comp\( \mathcal{G} \).composition-series-length-one)
hence carrier \( G = \{1_G\} \) by (metis comp\( \mathcal{G} \).composition-series-triv-group)
with length char\( \mathcal{G} \) have \( \mathcal{G} = \mathcal{H} \) using comp\( \mathcal{H} \).composition-series-triv-group

49
by simp
  thus thesis by simp
next
  — Second trivial case: $\mathfrak{G}$ is simple.
  assume length $\mathfrak{G} = 2$
  hence $\mathfrak{G} \text{char:} \mathfrak{G} = \{1_G\}$, carrier $G$ by (metis comp$\mathfrak{G}$.length-two-unique)
  hence simple: simple-group $G$ by (metis comp$\mathfrak{G}$.composition-series-simple-group)
  hence $\mathfrak{H} = \{1_G\}$, carrier $G$ using comp$\mathfrak{H}$.composition-series-simple-group
by auto
with $\mathfrak{G} \text{char}$ have $\mathfrak{G} = \mathfrak{H}$ by simp
thus thesis by simp
qed
next
case False
  — Non-trivial case: $\mathfrak{G}$ has length at least 3.
  hence length:length $\mathfrak{G} \geq 3$ by simp
  — First we show that $\mathfrak{H}$ must have a length of at least 3.
  hence $\neg$ simple-group $G$ using comp$\mathfrak{G}$.composition-series-simple-group by auto
  hence $\mathfrak{H} \neq \{1_G\}$, carrier $G$ using comp$\mathfrak{H}$.composition-series-simple-group by auto
  hence length $\mathfrak{H} \neq 2$ using comp$\mathfrak{H}$.length-two-unique by auto
moreover from length have carrier $G \neq \{1_G\}$ using comp$\mathfrak{G}$.composition-series-length-one
comp$\mathfrak{G}$.composition-series-triv-group by auto
  hence length $\mathfrak{H} \neq 1$ using comp$\mathfrak{H}$.composition-series-length-one comp$\mathfrak{H}$.composition-series-triv-group
by auto
  moreover from comp$\mathfrak{H}$.notempty have length $\mathfrak{H} \neq 0$ by simp
  ultimately have length$\mathfrak{H}$big:length $\mathfrak{H} \geq 3$ using comp$\mathfrak{H}$.notempty by arith
  def $m \equiv$ length $\mathfrak{H} - 1$
  def $n \equiv$ length $\mathfrak{G} - 1$
  from length$\mathfrak{H}$big have $m' : m > 0$ m < length $\mathfrak{H}$ $(m - 1) + 1 <$ length $\mathfrak{H}$ $m - 1 =$ length $\mathfrak{H} - 2 m - 1 + 1 =$ length $\mathfrak{H} - 1 m - 1 <$ length $\mathfrak{H}$
    unfolding m-def by auto
  from length have $n' : n > 0$ n < length $\mathfrak{G}$ $(n - 1) + 1 <$ length $\mathfrak{G}$ $n - 1 =$ length $\mathfrak{G} - 2 n - 1 + 1 =$ length $\mathfrak{G} - 1$ unfolding n-def by auto
  def $\mathfrak{G} \text{Pn} \equiv G(\text{carrier} := \mathfrak{H} ! (n - 1))$
  def $\mathfrak{H} \text{Pm} \equiv G(\text{carrier} := \mathfrak{H} ! (m - 1))$
  then interpret grp$\mathfrak{G}$.Pn: group $\mathfrak{G} \text{Pn}$ unfolding $\mathfrak{G} \text{Pn}$-def using n'-by (metis comp$\mathfrak{G}$.normal-series-subgroups comp$\mathfrak{G}$.subgroup-imp-group)
    interpret grp$\mathfrak{H}$.Pm: group $\mathfrak{H} \text{Pm}$ unfolding $\mathfrak{H} \text{Pm}$-def using m'-by (metis comp$\mathfrak{H}$.normal-series-subgroups comp$\mathfrak{H}$.subgroup-imp-group)
1 (2) group.subgroup-imp-group by force
  have finGBl:finite (carrier $\mathfrak{G} \text{Pn}$) using $\mathfrak{G}$ - 1 < length $\mathfrak{G}$, 1(3) unfolding
$\mathfrak{G} \text{Pn}$-def using comp$\mathfrak{G}$.normal-series-subgroups comp$\mathfrak{G}$.subgroup-finite by auto
  have finHBl:finite (carrier $\mathfrak{H} \text{Pm}$) using $\mathfrak{H}$ - 1 < length $\mathfrak{H}$, 1(3) unfolding
$\mathfrak{H} \text{Pm}$-def using comp$\mathfrak{H}$.normal-series-subgroups comp$\mathfrak{H}$.subgroup-finite by auto
  have quot$\mathfrak{G}$.notempty:comp$\mathfrak{G}$.quotients $\neq \emptyset$ using comp$\mathfrak{G}$.quotients-length
  length$\mathfrak{H}$big by auto
  have quot$\mathfrak{H}$.notempty:comp$\mathfrak{H}$.quotients $\neq \emptyset$ using comp$\mathfrak{H}$.quotients-length
  length$\mathfrak{H}$big by auto

50
— Instantiate truncated composition series since they are used for both cases

\texttt{interpret } \texttt{last butlast: composition-series } \texttt{Pm take m } \texttt{butlast using compP. composition-series-prefix-closed m' \{1,2\} Pm-def by auto}

\texttt{interpret } \texttt{butlast: composition-series } \texttt{Pn take n } \texttt{butlast using compP. composition-series-prefix-closed n' \{1,2\} Pn-def by auto}

\texttt{have taken:n = length (take n } \texttt{butlast using length-take n' \{2\} by auto}

\texttt{have taken:m = length (take m } \texttt{butlast using length-take m' \{2\} by auto}

\texttt{show } ?\texttt{thesis}

\texttt{proof (cases } \texttt{last butlast} \texttt{(! (m - 1) = } \texttt{butlast ! (n - 1))}

\texttt{— If } \texttt{last butlast} \texttt{(! (l - 1) = butlast ! 1, everything is simple...}

\texttt{case True}

\texttt{— The last quotients of } \texttt{butlast} \texttt{and } \texttt{last butlast} \texttt{are equal.}

\texttt{have } \texttt{lasteq: last compP.quotients = last compP.quotients}

\texttt{proof —}

\texttt{from length have by:length } \texttt{butlast ! - 1 - 1 + 1 = length } \texttt{butlast ! - 1 by (metis Suc-diff-1 Suc-eq-plus1 n' \{1\} n-def)}

\texttt{from length butlast big have by:length } \texttt{butlast ! - 1 - 1 + 1 = length } \texttt{butlast ! - 1 by (metis Suc-diff-1 Suc-eq-plus1 \{0 < m\} m-def)}

\texttt{have last compP.quotients = G Mod } \texttt{butlast ! (n - 1) using length compP.last-quotient unfolding n-def by auto}

\texttt{also have } \texttt{... = G Mod } \texttt{butlast ! (m - 1) using True by simp}

\texttt{also have } \texttt{... = last compP.quotients using length butlast big compP.last-quotient unfolding m-def by auto}

\texttt{finally show } ?\texttt{thesis .}

\texttt{qed}

\texttt{from taken have by: mset (map group.iso-class butlast.quotients) = mset (map group.iso-class } \texttt{butlast.quotients)}

\texttt{using } \texttt{? \texttt{(}1\texttt{) True n' \{5\} grpPn.is-group finGbl } \texttt{butlast.is-composition-series butlast.is-composition-series unfolding } \texttt{Pn-def butlast.Pm-def by metis}

\texttt{have mset (map group.iso-class compP.quotients)}

\texttt{= mset (map group.iso-class (butlast compP.quotients @ [last compP.quotients]) by (simp add: quotP.notempty)}

\texttt{also have } \texttt{... = mset (map group.iso-class (butlast.quotients @ [last (compP.quotients)]) using compP.quotients-butlast length unfolding n-def Pn-def by auto}

\texttt{also have } \texttt{... = mset (map group.iso-class butlast.quotients) @ [group.iso-class (last (compP.quotients)) #] by auto}

\texttt{also have } \texttt{... = mset (map group.iso-class butlast.quotients) + \{# group.iso-class (last (compP.quotients)) #\} by auto}

\texttt{also have } \texttt{... = mset (map group.iso-class butlast.quotients) + \{# group.iso-class (last (compP.quotients)) #\} using ind by simp}

\texttt{also have } \texttt{... = mset (map group.iso-class butlast.quotients) + \{# group.iso-class (last (compP.quotients)) #\} using lasteq by simp}

\texttt{also have } \texttt{... = mset (map group.iso-class butlast.quotients) @ [group.iso-class (last (compP.quotients)]) by auto}

\texttt{also have } \texttt{... = mset (map group.iso-class butlast.quotients) @ [last (compP.quotients)]) by auto}

\texttt{also have } \texttt{... = mset (map group.iso-class butlast.quotients) @ [last (compP.quotients)]) by auto}

\texttt{also have } \texttt{... = mset (map group.iso-class butlast.quotients) @ [last (compP.quotients)]) by auto}

\texttt{also have } \texttt{... = mset (map group.iso-class butlast.quotients) @ [last (compP.quotients)]) by auto}

\texttt{also have } \texttt{... = mset (map group.iso-class butlast.quotients) @ [last (compP.quotients)]) by auto}

\texttt{also have } \texttt{... = mset (map group.iso-class butlast.quotients) @ [last (compP.quotients)]) by auto
also have \( \ldots = \text{mset (map group.iso-class comp\(\delta\).quotients)} \) using append-butlast-last-id quotas\(\delta\)notempty by simp

finally show \( \text{thesis} \).

def \( \exists \text{Pm}\text{Int}\text{G} \equiv \text{G}[(\text{carrier} := \exists \text{)} (m - 1) \cap \text{G}! (n - 1)]\)

interpret \( \exists \text{Pmmax: max-normal-subgroup } \exists \text{G}! (n - 1) \) unfolding \( n\)-def
by (metis add-lessD1 diff-diff-add \( n'(3) \) add.commute one-add-one \( I(3) \)
comp\(\delta\).snd-to-last-max-normal)

interpret \( \exists \text{Pmmax: max-normal-subgroup } \exists \text{G}! (m - 1) \) unfolding \( m\)-def
by (metis add-lessD1 diff-diff-add \( m'(3) \) add.commute one-add-one \( I(3) \)
comp\(\delta\).snd-to-last-max-normal)

have \( \exists \text{PnmnormG} : \exists \text{G}! (m - 1) \triangleleft \text{G} \) using \( \text{comp\(\delta\).normal-series-snd-to-last } m'(4) \)
unfolding \( m\)-def by auto

have \( \exists \text{PnmnormG} : \exists \text{G}! (n - 1) \triangleleft \text{G} \) using \( \text{comp\(\delta\).normal-series-snd-to-last } n'(6) \)
unfolding \( n\)-def by auto

have \( \exists \text{Pmmin}\text{G} : \exists \text{G}! (m - 1) \cap \text{G}! (n - 1) \triangleleft \text{G} \) using \( \exists \text{PmnormG} \)
(\( \exists \text{G} \)\text{PnmnormG} by (rule \text{comp\(\delta\).normal-subgroup-intersect})

have \( \text{Intnorm}\text{G} : \text{G}! (m - 1) \cap \text{G}! (n - 1) \triangleleft \text{G} \) using \( \text{G} \)\text{PnmnormG} \( \exists \text{PmnormG} \)
(\( \exists \text{PmnormG} \)\text{Int-lower2 unfolding } \( \exists \text{Pn}\)-def
by (metis \text{comp\(\delta\).normal-restrict-supergroup comp\(\delta\).normal-series-subgroups comp\(\delta\).normal-subgroup-intersect } n'(4) \))

then interpret \( \text{grp}\text{G} \text{PnMod}\exists \text{Pm}\text{Int}\text{G} G: \text{group } \text{G} \)\text{Pn Mod } \exists \text{G}! (m - 1) \cap \( \exists \text{G}! (n - 1) \)
by (rule \text{normal.factorgroup-is-group})

have \( \text{Intnorm}\text{G} \exists \text{Pm}: \exists \text{G}! (m - 1) \cap \text{G}! (n - 1) \triangleleft \exists \text{Pm} \) using \( \exists \text{PmnormG} \)
(\( \exists \text{G} \)\text{PnmnormG} \text{Int-lower2 Int-commute unfolding } \exists \text{PmnormG} \)
by (metis \text{comp\(\delta\).normal-restrict-supergroup comp\(\delta\).normal-series-subgroups-intersect comp\(\delta\).normal-series-snd-to-last-\text{m}'(6))\)

then interpret \( \text{grp}\exists \text{PmMod}\exists \text{Pm}\text{Int}\text{G} \exists \text{G}! (m - 1) \cap \( \exists \text{G}! (n - 1) \)
by (rule \text{normal.factorgroup-is-group})

— Show that the second to last entries are not contained in each other.

have \( \neg \exists \text{PmSub}\text{G} \exists \text{Pn: } \neg (\exists \text{G}! (m - 1) \subseteq \text{G}! (n - 1)) \) using \( \exists \text{Pmmax}\.max-normal \)
(\( \exists \text{G} \)\text{PnmnormG} \text{False}[\text{symmetric}] \( \exists \text{G} \)\text{Pnmproper by simp}

have \( \neg \exists \text{PmSub}\text{G} \exists \text{Pn: } \neg (\exists \text{G}! (n - 1) \subseteq \exists \text{G}! (m - 1)) \) using \( \exists \text{Pmmax}\.max-normal \)
(\( \exists \text{G} \)\text{PnmnormG} \text{False } \( \exists \text{G} \)\text{Pnmproper by simp}

— Show that \( \text{G} \)\text{Mod } \exists \text{G}! (m - 1) \cap \( \exists \text{G}! (n - 1) \) is a simple group.

have \( \exists \text{PmSub}\text{Setmult}\exists \text{G}! (m - 1) \subseteq \exists \text{G}! (m - 1) \triangleleft \# > \text{G} \)\text{G}! (n - 1) \)
using \text{second-isomorphism-grp.H-contained-in-set-mult } \( \exists \text{G} \)\text{Pnmmax.is-normal}
(\( \exists \text{G} \)\text{PnmnormG} \text{normal-imp-subgroup unfolding } \text{second-isomorphism-grp-def second-isomorphism-grp-axioms-def}
\text{max-normal-subgroup-def by metis}

have \( \exists \text{G} \)\text{PnmSub}\text{Setmult}: \( \exists \text{G}! (n - 1) \subseteq \exists \text{G}! (m - 1) \triangleleft \# > \text{G} \)\text{G}! (n - 1) \)
using \text{second-isomorphism-grp.S-contained-in-set-mult } \( \exists \text{G} \)\text{Pnmmax.is-normal}
(\( \exists \text{G} \)\text{PnmnormG} \text{normal-imp-subgroup unfolding } \text{second-isomorphism-grp-def second-isomorphism-grp-axioms-def}

52
max-normal-subgroup-def by metis
have \(\mathcal{G}! (n - 1) \neq (\mathcal{G}! (n - 1)) \supseteq G (\mathcal{G}! (n - 1))\) using \(\mathcal{G}! PnSubSetmult\)
not\(\mathcal{G}! PnSub\mathcal{G} Pn\) by auto
hence set-mult\(G:(\mathcal{G}! (n - 1)) \supseteq G (\mathcal{G}! (n - 1)) = \text{carrier}\ G\)
using \(\mathcal{G}! Pnmax.max-normal \mathcal{G} Pnmax.is-normal \mathcal{G} PnmaxG \) comp\(\mathcal{G}.\)normal-subgroup-set-mult-closed
\(\mathcal{G}! PnSubSetmult\) by metis
then obtain \(\varphi\) where \(\varphi \in (\mathcal{G}! Pn Mod (\mathcal{G}! (m - 1) \cap \mathcal{G}! (n - 1))) \cong (G\{\text{carrier} := \text{carrier} G\} Mod \mathcal{G}! (m - 1))\)
using second-isomorphism-grp.normal-intersection-quotient-isom \(\mathcal{G}! PnmaxG\)
\(\mathcal{G}! PnmaxG.is-normal normal-imp-subgroup\)
unfolding second-isomorphism-grp-def second-isomorphism-grp-axioms-def
max-normal-subgroup-def \(\mathcal{G}! Pn-def\) by metis
hence \(\varphi: \varphi \in (\mathcal{G}! Pn Mod (\mathcal{G}! (m - 1) \cap \mathcal{G}! (n - 1))) \cong (G Mod \mathcal{G}! (m - 1)\)
by auto
then obtain \(\varphi 2\) where \(\varphi 2: \varphi 2 \in (G Mod \mathcal{G}! (m - 1) \cap \mathcal{G}! (n - 1)) \cong (\mathcal{G}! Pn Mod (\mathcal{G}! (m - 1) \cap \mathcal{G}! (n - 1)))\)
by auto
using group.iso-sym grp\(\mathcal{G}! PnMod\)\(\mathcal{G}! Pnint\)\(\mathcal{G}! Pn.is-group\) by auto
moreover have simple-group \((G\{\text{carrier} := \mathcal{G}! (m - 1 + 1)\} Mod \mathcal{G}! (m - 1))\) using comp\(\mathcal{G}.\).last_last_conv_nth
comp\(\mathcal{G}.\).not_empty m\(\{5\}\) by fastforce
ultimately have simple\(\mathcal{G}! PnModInt.simple-group\) \((\mathcal{G}! Pn Mod (\mathcal{G}! (m - 1) \cap \mathcal{G}! (n - 1)))\)
by auto
using simple-group.iso-simple grp\(\mathcal{G}! PnMod\)\(\mathcal{G}! Pnint\)\(\mathcal{G}! Pn.is-group\) by auto
interpret grp\(G Mod\)\(\mathcal{G}! Pn\): group \((G Mod \mathcal{G}! (m - 1))\) by (metis \(\mathcal{G}! PnmaxG\)
normal_factoprgroup-is-group)

— Show analogues of the previous statements for \(\mathcal{H}! (m - 1)\) instead of \(\mathcal{G}! (n - 1)\).

have \(\mathcal{H}! PnSubSetmult': \mathcal{H}! (m - 1) \subseteq \mathcal{G}! (n - 1) \supseteq G (\mathcal{G}! (n - 1))\)
using second-isomorphism-grp.S-contained-in-set-mult \(\mathcal{G}! PnmaxG.is-normal\)
\(\mathcal{G}! PnmaxG.normal-imp-subgroup\)
unfolding second-isomorphism-grp-def second-isomorphism-grp-axioms-def
max-normal-subgroup-def by metis
have \(\mathcal{G}! PnSubSetmult': \mathcal{G}! (n - 1) \subseteq \mathcal{G}! (n - 1) \supseteq G (\mathcal{G}! (n - 1))\)
using second-isomorphism-grp.H-contained-in-set-mult \(\mathcal{G}! PnmaxG.is-normal\)
\(\mathcal{G}! PnmaxG.normal-imp-subgroup\)
unfolding second-isomorphism-grp-def second-isomorphism-grp-axioms-def
max-normal-subgroup-def by metis
have \(\mathcal{H}! (m - 1) \neq (\mathcal{G}! (n - 1)) \supseteq G (\mathcal{G}! (n - 1))\) using \(\mathcal{G}! PnSubSetmult'\)
not\(\mathcal{G}! PnSub\mathcal{H}! Pn\) by auto
hence set-mult\(G:(\mathcal{G}! (n - 1)) \supseteq G (\mathcal{G}! (n - 1)) = \text{carrier}\ G\)
using \(\mathcal{G}! PnmaxG.max-normal \mathcal{G} PnmaxG.is-normal \mathcal{G} PnmaxG \) comp\(\mathcal{G}.\).normal-subgroup-set-mult-closed
\(\mathcal{G}! PnSubSetmult'\) by metis
from set-mult\(G\) obtain \(\psi\) where \(\psi \in (\mathcal{H}! Pm Mod (\mathcal{G}! (n - 1) \cap \mathcal{G}! (m - 1))) \cong (G\{\text{carrier} := \text{carrier}\ G\} Mod \mathcal{G}! (n - 1))\)
using second-isomorphism-grp.normal-intersection-quotient-isom \(\mathcal{G}! PnmaxG\)
\(\mathcal{G}! PnmaxG.is-normal normal-imp-subgroup\)
unfolding second-isomorphism-grp-def second-isomorphism-grp-axioms-def

53
max-normal-subgroup-def \( \mathcal{S} Pm-def \) by metis

hence \( \psi \psi \in (\mathcal{S} Pm \text{ Mod } (\mathcal{S} ! (m - 1) \cap (\mathcal{G} ! (n - 1))) \cong (G\{\text{carrier := carrier } G\} \text{ Mod } \mathcal{G} ! (n - 1)) \) using Int-commute by metis

then obtain \( \psi 2 \) where \( \psi 2 : \psi 2 \in (G \text{ Mod } \mathcal{G} ! (n - 1)) \cong (\mathcal{S} Pm \text{ Mod } (\mathcal{S} ! (m - 1) \cap \mathcal{G} ! (n - 1))) \)

using group.iso-sgm grp\( \mathcal{S} Pm \text{ Mod } \mathcal{S} Pm \text{ normal } \mathcal{G} \) by auto

moreover have simple-group \( (G\{\text{carrier := } \mathcal{G} ! (n - 1 + 1)\} \text{ Mod } \mathcal{G} ! (n - 1)) \) using comp.G.last conv nth comp.G.notempty \( n'(3) \) by fastforce

ultimately have simple-group \( \mathcal{S} Pm \text{ Mod } \text{Int}:\text{simple-group } (\mathcal{S} Pm \text{ Mod } (\mathcal{S} ! (m - 1) \cap \mathcal{G} ! (n - 1))) \)

using simple-group.iso-group grp\( \mathcal{S} Pm \text{ Mod } \mathcal{S} Pm \text{ normal } \mathcal{G} \) by auto

interpret grp.GMod.GP: group \( (G \text{ Mod } \mathcal{G} ! (n - 1)) \) by (metis \( \mathcal{G} \) Pn norm G normal.factor-group iso-group)

— Instantiate several composition series used to build up the equality of quotient multisets.

def \( R \equiv \text{rendups-adj } (\text{map } (\text{op } (\mathcal{S} ! (m - 1)))) \) \( \mathcal{G} \)
def \( \mathcal{L} \equiv \text{rendups-adj } (\text{map } (\text{op } (\mathcal{G} ! (n - 1)))) \) \( \mathcal{S} \)

interpret \( R \equiv \text{composition-series } \mathcal{S} Pm \ R \) using comp.G.intersect-normal \( 1(3) \)

\( \mathcal{S} Pm \text{ norm G unfolding } R-def \mathcal{S} Pm-def \) by auto

interpret \( \mathcal{L} \equiv \text{composition-series } \mathcal{G} Pn \ L \) using comp.G.intersect-normal \( 1(3) \)

\( \mathcal{S} Pn \text{ norm G unfolding } \mathcal{L}-def \mathcal{S} Pn-def \) by auto

— Apply the induction hypothesis on \( \mathcal{S} \text{ butlast } \) and \( \mathcal{L} \)

from \( n'(2) \) have Suc \( (\text{length } \text{take } n \mathcal{G}) \leq \text{length } \mathcal{G} \) by auto

hence multisets.Gbutlast.L.mset (map group.iso-class \( \mathcal{G} \) butlast.quotients) = mset (map group.iso-class \( \mathcal{L} \) quotients)

using 1.hyps grp.GPn.is-group \( \text{finGbl } \mathcal{G} \) butlast.is-composition-series \( \mathcal{L} \) is-composition-series by metis

hence \( \text{length } \mathcal{L} : n = \text{length } \mathcal{L} \) using \( \mathcal{G} \) butlast.quotients-length \( \mathcal{L} \) quotients-length

length-map size-mset take by metis

hence \( \text{length } \mathcal{L} : \text{length } \mathcal{L} > 1 \) \( \text{length } \mathcal{L} - 1 > 0 \) \( \text{length } \mathcal{L} - 1 \leq \text{length } \mathcal{L} \)

using \( n'(6) \) length by auto

have Integ.Lsndlast.H\( \bar{\omega} ! (m - 1) \cap \mathcal{G} ! (n - 1) = \mathcal{L} ! (\text{length } \mathcal{L} - 1 - 1) \) proof

have \( \text{length } \mathcal{L} - 1 - 1 + 1 < \text{length } \mathcal{L} \) using length.L' by auto

moreover have KGnotempty:(map \( (\text{op } (\mathcal{G} ! (n - 1)))) \) \( \mathcal{S} \) \( \neq [\] using comp.G.notempty by (metis Nil-is-map-cone)

ultimately obtain \( i \) where \( \vdash i + 1 < \text{length } \text{(map } (\text{op } (\mathcal{G} ! (n - 1)))) \) \( \mathcal{S} \)

\( \mathcal{L} ! (\text{length } \mathcal{L} - 1 - 1) = \text{(map } (\text{op } (\mathcal{G} ! (n - 1)))) \) \( \mathcal{S} \) \( \mathcal{L} ! (\text{length } \mathcal{L} - 1 - 1 + 1) = \text{(map } (\text{op } (\mathcal{G} ! (n - 1)))) \) \( \mathcal{S} \) \( i + 1 \)

using rendups-adj obtain adjacency unfolding \( \mathcal{L}-def \) by force

hence \( \mathcal{L} ! (\text{length } \mathcal{L} - 1 - 1) = \mathcal{S} ! i \cap \mathcal{G} ! (n - 1) \) \( \mathcal{L} ! (\text{length } \mathcal{L} - 1 - 1 + 1) = \mathcal{S} ! (i + 1) \cap \mathcal{G} ! (n - 1) \) by auto

hence \( \mathcal{L} ! (\text{length } \mathcal{L} - 1) = \mathcal{S} ! (i + 1) \cap \mathcal{G} ! (n - 1) \) using length.L'(2)
by \(\text{(metis Suc-diff-I Suc-eq-plus1)}\)

hence \(\emptyset\text{Pnsub}\emptyset\text{Pm} : \emptyset \oplus (n - 1) \subseteq \emptyset \oplus (i + 1)\) using \(L.\text{last} \emptyset.\text{notempty last-conv-nth unfolding } \emptyset.\text{Pn-def}\) by auto

from \(i(1)\) have \(i + 1 < m + 1\) unfolding \(m\)-def by auto

moreover have \(\neg (i + 1) \leq m - 1\) using \(\text{comp}\emptyset.\text{entries-mono } m'(6)\)

not\(\emptyset\text{Pnsub}\emptyset\text{Pm} \emptyset\text{Pnsub}\emptyset\text{Pm}\) by fastforce

ultimately have \(m - 1 = i\) by auto

with \(i\) show \(\text{thesis}\) by auto

qed

hence \(\text{last-conv-nth}\) unfolding \(\emptyset.\text{Pn-def}\) by auto

then interpret \(\emptyset.\text{butlast}\) : \(\text{composition-series}\emptyset\text{Pm} \emptyset\text{Int} \emptyset\text{Pn}\) take \((\text{length } L - 1)\) \(L\) using \(\emptyset.\text{composition-series-prefix-closed}\) by metis

from \((\text{length } L > 1)\) have \(\emptyset.\text{quotients}\emptyset.\text{notempty}\): \(\emptyset.\text{quotients} \neq []\) unfolding \(\emptyset.\text{quotients-def}\) by auto

---

Apply the induction hypothesis on \(\emptyset.\text{butlast}\) and \(\emptyset.\text{butlast}\)

have \(\text{length } \emptyset > 1\)

proof (rule ccontr)

assume \(\neg \text{length } \emptyset > 1\)

with \(\emptyset.\text{notempty}\) have \(\text{length } \emptyset = 1\) by \(\text{(metis One-nat-def Suc-less-I length-greater-0-def)}\)

hence carrier \(\emptyset\text{Pm} = \{1_{\emptyset\text{Pm}}\}\) using \(\emptyset.\text{composition-series-length-one } \emptyset.\text{composition-series-trive-group}\) by auto

hence carrier \(\emptyset\text{Pm} = \{1_{\emptyset}\}\) unfolding \(\emptyset.\text{Pm-def}\) by auto

hence carrier \(\emptyset\text{Pm} \subseteq \emptyset \oplus (n - 1)\) using \(\emptyset.\text{Pnmax.is-subgroup sub-group.one-closed}\) by auto

with \(\not\emptyset\text{PmSub}\emptyset\text{Pn}\) show \(\text{False}\) unfolding \(\emptyset.\text{Pm-def}\) by auto

qed

hence \(\text{length} \emptyset : \text{length } \emptyset - 1 > 0\) \(\text{length } \emptyset - 1 \leq \text{length } \emptyset\) by auto

have \(\text{Integ}R\text{ndlast}:\emptyset \oplus ((m - 1) \cap \emptyset) \oplus (n - 1) = \emptyset \oplus (\text{length } \emptyset - 1 - 1)\)

proof —

have \(\text{length } \emptyset - 1 - 1 + 1 < \text{length } \emptyset\) using \(\text{length}\emptyset.\text{R}\) by auto

moreover have \(\text{K}\emptyset.\text{notempty}:\text{(map } (\text{op } \cap (\emptyset) \oplus (m - 1))\) \(\emptyset) \neq []\) using \(\text{comp}\emptyset.\text{notempty}\) by \(\text{(metis Nil-is-map-one)}\)

ultimately obtain \(i\) where \(i:i + 1 < \text{length } (\text{map } (\text{op } \cap (\emptyset) \oplus (m - 1)))\)

\(\emptyset\)

\(\emptyset \oplus (\text{length } \emptyset - 1 - 1) = \text{map } (\text{op } \cap (\emptyset) \oplus (m - 1))\) \(\emptyset\) ! \(i \oplus \emptyset \oplus (m - 1) \emptyset \oplus (\text{length } \emptyset - 1 - 1 + 1) = \emptyset \oplus \emptyset \oplus (m - 1) \emptyset \oplus (\text{length } \emptyset - 1 - 1 + 1)\) ! \(i + 1\) \(\cap \emptyset\) ! \(m - 1\) by auto

hence \(\emptyset \oplus (\text{length } \emptyset - 1) = \emptyset \oplus (i + 1) \cap \emptyset \oplus (m - 1)\) using \(\text{length}\emptyset.\text{R}(1)\)

by \(\text{(metis Suc-diff-I Suc-eq-plus1)}\)

hence \(\emptyset\text{PmSub}\emptyset\text{Pn}:\emptyset \oplus (m - 1) \subseteq \emptyset \oplus (i + 1)\) using \(\emptyset.\text{last} \emptyset.\text{notempty last-conv-nth unfolding } \emptyset.\text{Pn-def}\) by auto

from \(i(1)\) have \(i + 1 < n + 1\) unfolding \(n\)-def by auto

moreover have \(\neg (i + 1) \leq n - 1\) using \(\text{comp}\emptyset.\text{entries-mono } n'(2)\)

not\(\emptyset\text{PmSub}\emptyset\text{Pn} \emptyset\text{PmSub}\emptyset\text{Pn}\) by fastforce
ultimately have $n - 1 = i$ by auto
with $i$ show thesis by auto
qed

have composition-series (G(carrier := #) ! (length # − 1)) (take (length # − 1) #)
  using length#'. #.composition-series-prefix-closed unfolding #PmInt#Pn-def
  #Pm-def by fastforce
  then interpret #butlast: composition-series #PmInt#Pn (take (length # − 1) #)
  using Inteqnd last unfolding #PmInt#Pn-def by auto
  from finGrbl have finInt: finite (carrier #PmInt#Pn) unfolding #PmInt#Pn-def
  #Pn-def by simp

  moreover have Suc (length (take (length # − 1) #)) ≤ length # using
  length# unfolding n-def using n'(2) by auto

  ultimately have multisets#butlast:mset (map group.iso-class #butlast.quotients)
    = mset (map group.iso-class #butlast.quotients)
  using 1.hyps #butlast.is-group #butlast.is-composition-series #butlast.is-composition-series
  by auto
  hence length (take (length # − 1) #) = length (take (length # − 1) #)
  using #butlast.quotients-length #butlast.quotients-length length-map size-mset
  by metis
  hence length (take (length # − 1) #) = n − 1 using length# n'(1) by auto

  hence length#length # = n by (metis Suc-diff-1 #.notempty butlast-conv-take
  length-butlast length-greater-0-conv n'(1))

  -- Apply the induction hypothesis on # and #butlast
  from Inteqnd last have #nd last #PmInt#Pn = (=#Pm(carrier := #) !
  (length # − 1 − 1)) unfolding #PmInt#Pn-def #Pm-def #.def by auto

  from length# have Suc (length #) ≤ length # using n'(2) by auto

  hence multisets#butlast:mset (map group.iso-class #butlast.quotients) =
    mset (map group.iso-class #.quotients)
  using 1.hyps grp#Pm.is-group finGrbl #butlast.is-composition-series #.is-composition-series
  by metis

  hence length#m = length # using #butlast.quotients-length #.quotients-length
  length-map size-mset ltake m by metis

  hence length # > 1 length # − 1 > 0 length # − 1 ≤ length # using m'(4)

  hence quotse#notempty #.quotients ≠ [] unfolding #.quotients-def by auto

  interpret #butlastadd#Pn: composition-series #Pn (take (length # − 1) #)
    @[# ! (n − 1)]
  using grp#Pn.composition-series-extend #butlast.is-composition-series simple#PnModInt Intnorm#Pn

  unfolding #Pn-def #PmInt#Pn-def by auto
  interpret #butlastadd#Pm: composition-series #Pm (take (length # − 1) #)
    @[# ! (m − 1)]
  using grp#Pm.composition-series-extend #butlast.is-composition-series simple#PmModInt Intnorm#Pm

  unfolding #Pm-def #PmInt#Pn-def by auto
As a corollary, we see that the composition series of a fixed group all have the same length.

\textbf{have} \ mset (\text{map group.iso-class comp} \mathfrak{G}.\text{quotients})
\begin{align*}
&= \ mset (\text{map group.iso-class } ((\text{butlast comp} \mathfrak{G}.\text{quotients}) \ @ \ [\text{last comp} \mathfrak{G}.\text{quotients}])) \text{ using } \text{quot} \mathfrak{G}\text{notempty by simp} \\
&\text{also have } \ldots = \ mset (\text{map group.iso-class } (\mathfrak{G}\text{butlast.quotients} \ @ \ [G \ Mod \ \mathfrak{G}] \! (n - 1)))) \\
&\text{using } \text{compG.quotients-butlast comp} \mathfrak{G}.\text{last-quotient length } \text{unfolding n-def} \\
&\mathfrak{G}Pn\text{-def by auto} \\
&\text{also have } \ldots = \ mset (\text{map group.iso-class } (\mathfrak{G}\text{butlast.quotients} \ @ \ [\mathfrak{G}Pn \ Mod \ \mathfrak{G}] \! (m - 1) \cap \mathfrak{G} ! (n - 1)))) + \{\# \text{ group.iso-class } (G \ Mod \ \mathfrak{G} ! (n - 1)) \#\} \\
&\text{using } \text{multisets}\mathfrak{G}\text{butlastL quot}\mathfrak{G}\text{notempty by simp} \\
&\text{also have } \ldots = \ mset (\text{map group.iso-class } (\mathfrak{G}\text{butlast.quotients} \ @ \ [\mathfrak{G}Pn \ Mod \ \mathfrak{G}] \! (m - 1) \cap \mathfrak{G} ! (n - 1)))) + \{\# \text{ group.iso-class } (G \ Mod \ \mathfrak{G} ! (n - 1)) \#\} \\
&\text{using } \text{L.quotients-butlast } \mathfrak{L}.\text{last-quotient } (\text{length } \mathfrak{L} > 1) \text{ Unfold } \text{last-quotient unfolding } n\text{-def by auto} \\
&\text{also have } \ldots = \ mset (\text{map group.iso-class } (\mathfrak{R}\text{butlast.quotients}) + \{\# \text{ group.iso-class } (\mathfrak{R}Pn \ Mod \ \mathfrak{R} ! (m - 1) \cap \mathfrak{R} ! (n - 1)) \#\} + \{\# \text{ group.iso-class } (G \ Mod \ \mathfrak{R} ! (n - 1)) \#\} \\
&\text{by } \text{metis} \\
&\text{also have } \ldots = \ mset (\text{map group.iso-class } ((\text{butlast } \mathfrak{R}.\text{quotients}) \ @ \ [\text{last } \mathfrak{R}.\text{quotients}])) + \{\# \text{ group.iso-class } (G \ Mod \ \mathfrak{R} ! (m - 1)) \#\} \\
&\text{using } \mathfrak{R}.\text{quotients-butlast } \mathfrak{R}.\text{last-quotient } (\text{length } \mathfrak{R} > 1) \text{ Unfold } \text{last-quotient unfolding } m\text{-def by auto} \\
&\text{also have } \ldots = \ mset (\text{map group.iso-class } (\mathfrak{R}\text{butlast.quotients}) + \{\# \text{ group.iso-class } (G \ Mod \ \mathfrak{R} ! (m - 1)) \#\} \\
&\text{using } \text{multisets}\mathfrak{R}\text{butlastR quot}\mathfrak{R}\text{notempty by simp} \\
&\text{also have } \ldots = \ mset (\text{map group.iso-class } ((\text{butlast comp} \mathfrak{R}.\text{quotients}) \ @ \ [\text{last comp} \mathfrak{R}.\text{quotients}])) \\
&\text{using } \text{compR.quotients-butlast comp} \mathfrak{R}.\text{last-quotient length}\mathfrak{R}\text{big unfolding } m\text{-def by auto} \\
&\text{also have } \ldots = \ mset (\text{map group.iso-class } (\text{butlast comp} \mathfrak{R}.\text{quotients} \ @ \ [\text{last comp} \mathfrak{R}.\text{quotients}])) \\
&\text{using } \text{quot}\mathfrak{R}\text{notempty by simp} \\
&\text{finally show } \text{thesis}. \\
&\text{qed} \\
&\text{qed}
\end{align*}
corollary (in jordan-hoelder) jordan-hoelder-size:
shows length ⋀ = length ⋁
proof –
  have length ⋀ = length comp⋄.quotients + 1 by (metis comp⋄.quotients-length)
  also have ... = length (map group.iso-class comp⋄.quotients) + 1 by (metis length-map)
  also have ... = size (mset (map group.iso-class comp⋄.quotients)) + 1 by (metis size-mset)
  using jordan-hoelder-multisets is-group finite is-composition-series comp⋄.is-composition-series
  also have ... = length (map group.iso-class comp⋄.quotients) + 1 by (metis size-mset)
  also have ... = length comp⋄.quotients + 1 by (metis length-map)
  also have ... = length ⋁ by (metis comp⋄.quotients-length)
finally show ?thesis.
qed

end

References