The Jordan-Hölder Theorem

Jakob von Raumer

June 11, 2019

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 9

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 25
   8.1 Preliminaries ........................................... 25
   8.2 Normal Series .......................................... 27
   8.3 Composition Series .................................... 32

9 Isomorphism Classes of Groups 46
10 The Jordan-Hölder Theorem

theory SndIsomorphismGrp
imports
  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.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

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

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

lemma (in group) self-factor-iso:
  shows \((\lambda X. \text{the-elem } ((\lambda x. 1) \circ X)) \in \text{iso } (G \circ \text{Mod } \text{carrier } G) (G[\text{carrier } := \{1\}])\)
proof —
  have \(G[\text{carrier } := \{1\}]\) by (metis subgroup-imp-group triv-subgroup)
  hence group-hom \(G[\text{carrier } := \{1\}]\) \((\lambda x. 1)\) unfolding group-hom-def
  group-hom-axioms-def hom-def using is-group by auto
  moreover have \((\lambda x. 1) \circ \text{carrier } G = \text{carrier } (G[\text{carrier } := \{1\}])\) by auto
  moreover have kernel \(G\) \((G[\text{carrier } := \{1\}])\) \((\lambda x. 1) = \text{carrier } G\) unfolding kernel-def by auto
  ultimately show ?thesis using group-hom.FactGroup-iso-set 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 \(G\)

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

context second-isomorphism-grp
begin

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

lemma normal-subgrp-intersection-normal:
  shows \(S \cap H \triangleq (G[\text{carrier } := \{1\}])\)
proof (auto simp: groupS.normal-inv-iff)
  from subgrpS is-subgroup have \(\bigwedge x. x \in \{S, H\} \implies \text{subgroup } x G\) by auto
  hence subgroup \((\bigcap \{S, H\}) G\) using subgroups-Inter by blast
  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
next
fix \(g, h\)
assume \(g \in S\) and \(hH: h \in H\) and \(hS: h \in S\) \{ from \(g \in H\) subgrpS show \(g \odot h \odot \text{inv}_{G[\text{carrier } := \{1\}]} g \in H\) by (metis inv-op-closed2 subgroup.mem-carrier m-inv-consistent) \}
\[
\begin{align*}
&\{\text{from } g\, h S \text{ subgroup show } g \otimes h \otimes \text{inv } G(\text{carrier} := S)\} \ g \in S \text{ by (metis subgroup.m-closed subgroup.m-inv-closed m-inv-consistent)} \\
&\text{qed}
\end{align*}
\]

**lemma normal-set-mult-subgroup:**

- shows subgroup \((H <\#> S) G\)
- **proof** (rule subgroup)
  - show \(H <\#> S \subseteq \text{carrier } G\) by (metis setmult-subset-G subgroup.subset subgroup)
  - **next**
    - have \(1 \in H\) \(1 \in S\) using is-subgroup subgroup one-closed by auto
    - hence \(1 \otimes 1 \in H <\#> S\) unfolding set-mult-def by blast
    - thus \(H <\#> S \neq \{\}\) by auto
  - **next**
    - fix \(g\)
      - assume \(g : g \in H <\#> S\)
      - then obtain \(h s\) where \(h : h \in H\) and \(s : s \in S\) and \(ghs : g = h \otimes s\) unfolding set-mult-def by auto
      - hence \(s \in \text{carrier } G\) by (metis subgroup.mem-carrier subgroup)
      - with \(ghs\) obtain \(h'\) where \(h' : h' \in H\) and \(g = s \otimes h'\) using coset-eq unfolding r-coset-def l-coset-def by auto
      - with \(s\) have \(\text{inv } g = (\text{inv } h') \otimes (\text{inv } s)\) by (metis inv-mult-group mem-carrier subgroup mem-carrier subgroup)
      - moreover from \(h' S\) subgroup have \(inv h' \in H\) \(inv s \in S\) using subgroup.m-inv-closed m-inv-closed by auto
      - ultimately show \(\text{inv } g \in H <\#> S\) unfolding set-mult-def by auto
  - **next**
    - fix \(g \, g'\)
      - assume \(g : g \in H <\#> S\) and \(h : g' \in H <\#> S\)
      - then obtain \(h h' s \, s'\) where \(hh' s : h \in H h' \in H s \in S s' \in S\) and \(g = h \otimes s\) and \(g' = h' \otimes s'\) unfolding set-mult-def by auto
      - hence \(g \otimes g' = (h \otimes s) \otimes (h' \otimes s')\) by metis
      - also from \(hh' s s'\) have \(\text{inG}: h \in \text{carrier } G h' \in \text{carrier } G s \in \text{carrier } G s' \in \text{carrier } G\) using subgroup mem-carrier subgroup mem-carrier by force+
      - hence \((h \otimes s) \otimes (h' \otimes s') = h \otimes (s \otimes h') \otimes s'\) using m-assoc by auto
      - also from \(\text{hh' ss'}\) have \(h''\) where \(h'' : h'' \in H\) and \(s \otimes h' = h'' \otimes s\) using coset-eq unfolding r-coset-def l-coset-def by fastforce
      - hence \((h \otimes s) \otimes (h' \otimes s') = h \otimes (h'' \otimes s) \otimes s'\) by simp
      - also from \(h''\) have \(\text{inG} : h''\) using m-assoc mem-carrier by auto
      - finally have \(g \otimes g' = h \otimes h'' \otimes (s \otimes s')\).
      - moreover with \(h''\) \(hh' ss'\) have \(\ldots \in H <\#> S\) unfolding set-mult-def using subgroup subgroup m-closed by fastforce
      - ultimately show \(g \otimes g' \in H <\#> S\) by simp
      - **qed**

**lemma oneH:** \(1 \in H\) by (metis is-subgroup subgroup one-closed)
lemma \( \text{H-contained-in-set-mult} \):
\begin{align*}
\text{shows} & \quad H \subseteq H \triangleleft S \\
\text{proof (auto \ldots)} & \\
\text{have} & \quad 1 \in S \text{ by (metis subgroup.one-closed subgpS)} \\
\text{fix} & \quad x \\
\text{assume} & \quad x \in H \\
\text{with} & \quad 1 \in S \text{ unfolding set-mult-def by force} \\
\text{with} & \quad x \text{ show } x \in H \triangleleft S \text{ by (metis mem-carrier r-one)} \\
\text{qed}
\end{align*}

lemma \( \text{S-contained-in-set-mult} \):
\begin{align*}
\text{shows} & \quad S \subseteq H \triangleleft S \\
\text{proof (auto \ldots)} & \\
\text{fix} & \quad s \\
\text{assume} & \quad s \in S \\
\text{with} & \quad 1 \text{ unfolding set-mult-def by force} \\
\text{with} & \quad s \text{ show } s \in H \triangleleft S \text{ using subgpS subgroup.mem-carrier l-one by force} \\
\text{qed}
\end{align*}

lemma \( \text{normal-intersection-hom} \):
\begin{align*}
\text{shows} & \quad \text{group-hom}((G|\text{carrier} := S))((G|\text{carrier} := H \triangleleft S))\text{ Mod } H \quad (\lambda g. H \triangleleft g) \\
\text{proof (auto del: equalityI simp: group-hom-def group-hom-axioms-def hom-def groupS.is-group \ldots)} & \\
\text{have} & \quad \text{gr} \\
\text{moreover have} & \quad H \subseteq H \triangleleft S \text{ by (rule H-contained-in-set-mult)} \\
\text{moreover have} & \quad \text{subgroup } (H \triangleleft S) \text{ by (metis normal-set-mult-subgroup)} \\
\text{ultimately have} & \quad H \triangleleft (G|\text{carrier} := H \triangleleft S) \text{ using normal-restrict-supergroup by (metis inv-op-closed2 is-subgroup normal-inv-iff)} \\
\text{with} & \quad \text{gr} \text{ show } \text{group}((G|\text{carrier} := H \triangleleft S))\text{ Mod } H \text{ by (metis normal.factorgroup-is-group)} \\
\text{next} & \\
\text{fix} & \quad g \\
\text{assume} & \quad g \in S \\
\text{with} & \quad \text{subgpS have} 1 \otimes g \in H \triangleleft S \text{ unfolding set-mult-def by fastforce} \\
\text{with} & \quad g \text{ have } g \in H \triangleleft S \text{ by (metis l-one subgroup.mem-carrier subgpS)} \\
\text{thus} & \quad H \triangleleft g \in \text{carrier}((G|\text{carrier} := H \triangleleft S))\text{ Mod } H \text{ unfolding FactGroup-def RCOSETS-def r-coset-def by auto} \\
\text{next} & \\
\text{show} & \quad \forall x y. (x \in S; y \in S) \implies H \triangleleft x \otimes y = H \triangleleft x \triangleleft (H \triangleleft y) \\
\text{using} & \quad \text{normal.rcos-sum normal-axioms subgroup.mem-carrier subgpS by fastforce} \\
\text{qed}
\end{align*}

lemma \( \text{normal-intersection-hom-kernel} \):
\begin{align*}
\text{shows} & \quad \text{kernel}((G|\text{carrier} := S))((G|\text{carrier} := H \triangleleft S))\text{ Mod } H \quad (\lambda g. H \triangleleft g) = H \cap S \\
\text{proof -} & \\
\text{have} & \quad \text{kernel}((G|\text{carrier} := S))((G|\text{carrier} := H \triangleleft S))\text{ Mod } H \quad (\lambda g. H \triangleleft g) \\
\text{qed}
\end{align*}
lemma normal-intersection-hom-surj:
  shows \((\lambda g. H > g) \cdot carrier (G(|carrier := H > S|)) = carrier ((G(|carrier := H < S|)) Mod H)\)
proof auto
  fix \(g\)
  assume \(g \in S\)
  hence \(g \in H < S\) using S-contained-in-set-mult by auto
  thus \(H > g \in carrier ((G(|carrier := H < S|)) Mod H)\) unfolding FactGroup-def RCOSETS-def r-coset-def by auto
next
  fix \(x\)
  assume \(x \in carrier (G(|carrier := H < S|) Mod H)\)
  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 subgroup.mem-subset subset)
  also have \(...) H > s\ by (metis h is-group rcos-const)
  finally have \(x = H > s\).
  with \(s\) show \(x \in (>_ S) H \cdot S\) by simp
qed

Finally we can prove the actual isomorphism theorem:

theorem normal-intersection-quotient-isom:
  shows \((\lambda X.\ the-elem (\lambda g. H > g) \cdot X)) \in iso ((G(|carrier := S|)) Mod (H \cap S)) ((G(|carrier := H < S|)) Mod H)\)
2 Preliminary lemmas

A group of order 1 is always the trivial group.

**lemma** (in group) order-one-triv-iff:

shows \( (\text{order } G = 1) = (\text{carrier } G = \{1\}) \)

**proof**

assume \( \text{order } G = 1 \)
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
with \( x \) show \( \text{carrier } G = \{1\} \) by simp
next
assume \( \text{carrier } G = \{1\} \)
thus \( \text{order } G = 1 \) unfolding order-def by auto
qed

**lemma** (in group) finite-pos-order:

assumes \( \text{finite } (\text{carrier } G) \)
shows \( 0 < \text{order } G \)

**proof**

from one-closed finite show \( \lnot \)thesis unfolding order-def by (metis card-gt-0-iff subgroup-nonempty subgroup-self)
qed

**lemma** iso-order-closed:

assumes \( \phi \in \text{iso } G H \)
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 \( \text{subgroup } U (G | \text{carrier} := H) \)
shows \( U \subseteq H \)
using assms subgroup subset by force

**lemma** (in subgroup) subgroup-of-subgroup:

assumes \( \text{group } G \)
assumes \( \text{subgroup } U (G | \text{carrier} := H) \)
shows \( \text{subgroup } U G \)

**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 \textit{assms} a show \(x \otimes y \in U\) by (metis subgroup.m-closed)

next
  have \(1_{\textit{G}}\) \textit{carrier} := \(H\) \(= 1\) by simp
  with \textit{assms} show \(1 \in U\) by (metis subgroup.one-closed)

next
  have subgroup \(H \subseteq G\).
  fix \(x\)
  assume \(x \in U\) with \textit{assms} (2)
  have \(\text{inv}_{\textit{G}}(\text{carrier} := H) \times = \text{inv}_{\textit{G}}(\text{carrier} := H)\) \(x = \text{inv} x\) by (rule subgroup.m-inv-consistent)

moreover from \textit{assms} \((x \in U)\) have \(x \in H\) by (metis in-mono subgroup-of-restricted-group)
  with \textit{assms} (1) (subgroup \(H \subseteq G\) have \(\text{inv}_{\textit{G}}(\text{carrier} := H) \times = \text{inv} x\) by (rule group.m-inv-consistent)

ultimately show \(\text{inv} x \in U\) by simp

qed

Being a subgroup is preserved by surjective homomorphisms

\textbf{lemma} (in subgroup) \textbf{surj-hom-subgroup}:
  \textit{assumes} \(\varphi: \textit{group-hom} \ G \ F \ \varphi\)
  \textit{assumes} \(\varphi\text{surj} : \varphi'(\text{carrier } G) = \text{carrier } F\)
  \textit{shows} subgroup \((\varphi' \cdot H) F\)

\textit{proof}
  from \(\varphi\text{surj}\) show img-subset; \(\varphi' \subseteq \text{carrier } F\) unfolding iso-def bij-betw-def by auto

next
  fix \(f f'\)
  assume \(h : f \in \varphi' \cdot H\) and \(h' : f' \in \varphi' \cdot H\)
  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
  hence \(g \otimes_G g' \in H\) by (metis m-closed)
  hence \(\varphi(g \otimes_G g') \in \varphi' \cdot H\) by simp
  with \(g g'\) \(\text{show } f \otimes_F f' \in \varphi' \cdot H\) using group-hom.hom-mult by fastforce

next
  have \(\varphi 1 \in \varphi' \cdot H\) by auto
  with \(\varphi\) show \(1_F \in \varphi' \cdot H\) by (metis group-hom.hom-one)

next
  fix \(f\)
  assume \(f : f \in \varphi' \cdot H\)
  then obtain \(g\) where \(g : g \in H f = \varphi g\) by auto
  hence inv \(g \in H\) by auto
  hence \(\varphi(\text{inv} g) \in \varphi' \cdot H\) by auto
  with \(\varphi g\) subset show invF \(f \in \varphi' \cdot H\) using group-hom.hom-inv by fastforce

qed

... and thus of course by isomorphisms of groups.

\textbf{lemma} iso-subgroup:
  \textit{assumes} \(\text{groups:} \textit{group} \ G \ \textit{group} F\)
  \textit{assumes} \(HG: \textit{subgroup} \ H G\)
  \textit{assumes} \(\varphi:\varphi \in \textit{iso } G F\)
  \textit{shows} subgroup \((\varphi' \cdot H) F\)
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 ∈ iso G F
  shows (restrict φ H) ∈ iso (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 ∩ G
by (metis coset-eq is-subgroup normal)

Being a normal subgroup is preserved by surjective homomorphisms.

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

Being a normal subgroup is preserved by group isomorphisms.

lemma iso-normal-subgroup:
assumes groups:group G group F
assumes $\varphi: \varphi \in \text{iso } G F$
shows $(\varphi \cdot H) \triangleleft F$
proof --
from groups $\varphi$ have group-hom G F $\varphi$ 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 $\varphi H$
ultimately show $\text{thesis } \varphi$ unfolding 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
    using lagrange by blast
  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 ⊆ carrier G; K ⊆ carrier G; x ∈ carrier G ]
  =⇒ (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 ∀x∈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 assyms(1,2) normal-inv-iff
    setmult-rcos-assoc subgroup subset x)

  also have ... = M <#> (x <# N) by (metis assyms(2) normal.coset-eq x)

  also have ... = (M #> x) <#> N by (metis assyms(1,2) normal-imp-subgroup
    rcos-assoc-lcos subgroup subset x)

  also have ... = (x <# M) <#> N by (metis assyms(1) normal.coset-eq x)

  also have ... = x <# (M <#> N) by (metis assyms(1,2) normal-imp-subgroup
    setmult-lcos-assoc subgroup subset x)

  finally show M <#> N #> x = x <# (M <#> N).

**qed**

**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 $x : x \in \text{carrier } (G|\text{carrier} := H)$ $y \in \text{carrier } (G|\text{carrier} := H)$ 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.\text{one-closed} by fastforce
- next
  - fix $x$
  - assume $x \in H$
  - hence $x : x \in \text{carrier } (G|\text{carrier} := H)$ by auto
  - hence $\text{inv } G|\text{carrier} := H) x \in \text{carrier } (G|\text{carrier} := H)$ using assms(2) group.inv-closed by fastforce
  - hence $\text{inv } G|\text{carrier} := H) x \in \text{carrier } G$ using $x$ assms(1) by auto
  - moreover have $\text{inv } G|\text{carrier} := H) x \otimes x = 1$ using assms(2) group.\text{l-inv } $x$
    by fastforce
  - moreover have $x \in \text{carrier } G$ using $x$ assms(1) by auto
  - ultimately have $\text{inv } G|\text{carrier} := H) x = \text{inv } x$ using $\text{inv-equality[ symmetric]}$
    by auto
  - thus $\text{inv } x \in H$ 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 $N < G$ and $N \subseteq H$ and subgroup $H G$
- shows subgroup $(\text{rcosets } G|\text{carrier} := H) N)$ $(G \text{ Mod } N)$

**Proof**
- interpret $G\text{Mod} N$: group $G$ Mod $N$ using assms(1) by (rule normal.factorgroup-is-group)
- have $N < G|\text{carrier} := H$) using assms by (metis normal-restrict-supergroup)
- hence $\text{grpHN}: group (G|\text{carrier} := H) \text{ Mod } N)$ by (rule normal.factorgroup-is-group)
- have $(<#> G|\text{carrier} := H) = (\lambda U K. \{ h \in U \cup k \in K. \{ h \otimes G|\text{carrier} := H \}) k))$)
  using set-mult-def by metis
- moreover have $\ldots = (\lambda U K. \{ h \in U \cup k \in K. \{ h \otimes G\text{carrier} := H \}) k))$) by auto
- moreover have $(<#>) = (\lambda U K. \{ h \in U \cup k \in K. \{ h \otimes k \})$) using set-mult-def by metis
- ultimately have $(<#> G|\text{carrier} := H) = (\#) G$ by simp
- with $\text{grpHN}$ have group ($(G \text{ Mod } N)|\text{carrier} := (\text{rcosets } G|\text{carrier} := H) N))$
- unfolding $\text{FactGroup-def} by auto$
- moreover have $(\text{rcosets } G|\text{carrier} := H) N \subseteq \text{carrier } (G \text{ Mod } N)$ unfolding $\text{FactGroup-def} \text{ RCOSETS-def} r-coset-def$
using assms(3) subgroup.subset by fastforce
ultimately show \(?thesis\) using GModN.is-group group.restrict-group-imp-subgroup
by auto
qed

A normality relation survives factoring by a normal subgroup.

lemma (in group) normality-factorization:
assumes \(NG:N \triangleleft G\) and \(NH:N \subseteq H\) and \(HG:H \triangleleft G\)
shows \((rcosets_{G\{\text{carrier := } H\}} N) < (G \mod N)\)
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 \((rcosets_{G\{\text{carrier := } H\}} N) (G \mod N)\) using
  normal-imp-subgroup normal-subgroup-factorize by force
  next
  fix \(U\) \(V\)
  assume \(U:U \in \text{carrier } (G \mod N)\) and \(V:V \in rcosets_{G\{\text{carrier := } H\}} N\)
  then obtain \(g\) where \(g:g \in \text{carrier } G\ U = N \triangleright g\) unfolding FactGroup-def
  RCOSETS-def by auto
  from \(V\) obtain \(h\) where \(h:h \in H\ V = N \triangleright h\) unfolding FactGroup-def
  RCOSETS-def r-coset-def by auto
  hence \(hG:h \in \text{carrier } G\) using \(HG\) normal-imp-subgroup subgroup.mem-carrier
  by force
  hence \(ghG:g \otimes h \in \text{carrier } G\) using \(g\) m-closed by auto
  from \(g\) \(h\) have \(g \otimes h \otimes \text{inv } g \in H\) using \(HG\) normal-inv-iff by auto
  moreover have \(U <\#> V <\#> \text{inv } G \mod N\ U = N \triangleright (g \otimes h \otimes \text{inv } g)\)
  proof
    from \(g\) \(U\) have \(\text{inv } G \mod N\ U = N \triangleright \text{inv } g\) using \(NG\) normal.inv-FactGroup
    normal.rcos-inv by fastforce
    hence \(U <\#> V <\#> \text{inv } G \mod N\ U = (N \triangleright g) <\#> (N \triangleright h) <\#> (N \triangleright \text{inv } g)\)
    using \(g\) \(h\) by simp
    also have \(... = N \triangleright (g \otimes h) <\#> (N \triangleright \text{inv } g)\) using \(g\) \(hG\) \(NG\)
    normal.rcos-sum by force
    also have \(... = N \triangleright (g \otimes h \otimes \text{inv } g)\) using \(g\) inv-closed \(ghG\) \(NG\)
    normal.rcos-sum by force
    finally show \(?thesis\) .
  qed
  ultimately show \(U <\#> V <\#> \text{inv } G \mod N\ U \in rcosets_{G\{\text{carrier := } H\}} N\)
  unfolding RCOSETS-def r-coset-def by auto
qed
qed

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 \((\text{carrier } G)\)
shows \((\text{carrier } (G \mod H) = \{1_{G \mod H}\}) = (H = \text{carrier } G)\)
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 \(H = \text{carrier } G\) using subgroup.subset is-subgroup assms card-subset-eq unfolding order-def
    by metis
next
  from assms have ordrgt0:order \(G\) > 0 unfolding order-def by (metis subgroup.finite-imp-card-positive subgroup-self)
  assume \(H = \text{carrier } G\)
  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 ordrgt0 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 \((\bigcup A)\) = \(\{x \in \text{carrier } G. \ H \geq x \in A\}\)
proof
  show \((\bigcup A)\) \subseteq \(\{x \in \text{carrier } G. \ H \geq x \in A\}\)
  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 \(xx:x \in \text{carrier } G\) using subgroup.subset unfolding FactGroup-def RCOSETS-def r-coset-def by force
      from assms \(a\) obtain \(y\) where \(y:y \in \text{carrier } G \ a = H \geq y\) using subgroup.subset unfolding FactGroup-def RCOSETS-def by force
        with \(a\) have \(x \in H \geq y\) by simp
          hence \(H \geq y = H \geq x\) using \(y\) is-subgroup repr-independence by auto
            with \(y(2)\) \(a(1)\) have \(H \geq x \in A\) by auto
              with \(xx\) show \(x \in \{x \in \text{carrier } G. \ H \geq x \in A\}\) by simp
  qed
next
show \( \{ x \in \text{carrier } G. \ H \not\supseteq x \in A \} \subseteq \bigcup A \)

proof

fix \( x \)

assume \( \{ x \in \text{carrier } G. \ H \not\supseteq x \in A \} \)

hence \( xx \in \text{carrier } G \not\supseteq x \in A \) by auto

moreover have \( x \in H \not\supseteq x \) by (metis is-subgroup rcos-self xx (1))

ultimately show \( x \in \bigcup A \) by auto

qed

proof

have subgroup \( \{ x \in \text{carrier } G. \ H \not\supseteq x \in A \} \subseteq \text{carrier } G \) by auto

next

fix \( x \) \( y \)

assume \( \{ x \in \text{carrier } G. \ H \not\supseteq x \in A \} \) and \( y \in \{ x \in \text{carrier } G. \ H \not\supseteq x \in A \} \)

hence \( xx \in \text{carrier } G \not\supseteq x \in A \) and \( yy \in \text{carrier } G \not\supseteq y \in A \) by auto

hence \( xy \in \text{carrier } G \) by (metis m-closed)

from assms \( x \) \( y \) have \( (H \not\supseteq x) \not\supseteq (H \not\supseteq y) \in A \) using subgroup.m-closed

unfolding FactGroup-def by fastforce

hence \( H \not\supseteq (x \otimes y) \in A \) by (metis rcos-sum \( x(1) \) \( y(1) \))

with \( xy \) show \( x \otimes y \in \{ x \in \text{carrier } G. \ H \not\supseteq x \in A \} \) by simp

next

have \( H \not\supseteq 1 \in A \) using assms subgroup.one-closed unfolding FactGroup-def

by (metis coset-mult-one monoid.select-convs (2) subset)

with assms one-closed show \( 1 \in \{ x \in \text{carrier } G. \ H \not\supseteq x \in A \} \) by simp

next

fix \( x \)

assume \( \{ x \in \text{carrier } G. \ H \not\supseteq x \in A \} \)

hence \( xx \in \text{carrier } G \not\supseteq x \in A \) by auto

hence \( invx \in x \in \text{carrier } G \) using inv-closed by simp

from assms \( x \) have \( set-inv \) \( (H \not\supseteq x) \in A \) using subgroup.m-inv-closed by (metis inv-FactGroup subgroup.mem-carrier)

hence \( H \not\supseteq (invx) \in A \) by (metis rcos-inv \( x(1) \))

with \( invx \) show \( invx \in \{ x \in \text{carrier } G. \ H \not\supseteq x \in A \} \) by simp

qed

with assms factgroup-subgroup-union-char show ?thesis by auto

qed

lemma (in normal) factgroup-subgroup-union-normal:

assumes \( A \triangleleft (G \text{ Mod } H) \)

shows \( \bigcup A \triangleleft G \)

proof


have \( \{ x \in \text{carrier } G. \ H \#> x \in A \} \triangleleft G \)

unfolding normal-def normal-axioms-def

proof auto

from assms show subgroup \( \{ x \in \text{carrier } G. \ H \#> x \in A \} \) \( G \)

by (metis (full-types) factgroup-subgroup-union-char factgroup-subgroup-union-subgroup normal-imp-subgroup)

next

interpret Anormal: normal A (G Mod H) using assms by simp

fix x y

assume \( x : x \in \text{carrier } G \ y \in \{ x \in \text{carrier } G. \ H \#> x \in A \} \#> x \)

then obtain \( x' \) where \( x' \in \{ x \in \text{carrier } G. \ H \#> x \in A \} \ y = x' \odot x \)

unfolding r-coset-def by auto

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

from x(1) have Hz: H \#> x \in carrier (G Mod H) unfolding FactGroup-def RCOSETS-def by force

with \( x' \) have \( \text{inv } G \text{ Mod } H ((H \#> x)) \odot G \text{ Mod } H ((H \#> x')) \odot G \text{ Mod } H ((H \#> x)) \) \( A \) using Anormal.inv-op-closed1 by auto

hence (set-inv \( (H \#> x) \)) \( \#> \) (\( H \#> x' \)) \( \#> \) (\( H \#> x \)) \( A \) using inv-FactGroup Hz unfolding FactGroup-def by auto

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

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

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

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

ultimately have \( \text{inv } x \odot x' \odot x \in \{ x \in \text{carrier } G. \ H \#> x \in A \} \)

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

unfolding l-coset-def using x(1) by auto

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

also have \( \ldots = x' \odot x \) by (metis l-one r-inv x'(1) x(1))

also have \( \ldots = y \) by (metis y = x' \odot x)

finally have \( x \odot (\text{inv } x \odot x' \odot x) = y \).

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

next

interpret Anormal: normal A (G Mod H) using assms by simp

fix x y

assume \( x : x \in \text{carrier } G \ y \in x \#> \{ x \in \text{carrier } G. \ H \#> x \in A \} \)

then obtain \( x' \) where \( x' \in \{ x \in \text{carrier } G. \ H \#> x \in A \} \ y = x \odot x' \)

unfolding l-coset-def by auto

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

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

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

with \( x' \) have \( \text{inv } G \text{ Mod } H ((H \#> \text{inv } x)) \odot G \text{ Mod } H ((H \#> x')) \odot G \text{ Mod } H ((H \#> \text{inv } x)) \) \( A \) using inx Anormal.inv-op-closed1 by auto

hence (set-inv \( (H \#> \text{inv } x)) \#> (H \#> x') \#> (H \#> \text{inv } x) \) \( A \)
using inv-FactGroup Hinvx unfolding FactGroup-def by auto
    hence (H ≠> inv (inv x)) ◁▷ (H ≠> x') ◁▷ (H ≠> inv x) ∈ A using
    invx by (metis rcos-inv)
    hence (H ≠> x) ◁▷ (H ≠> x') ◁▷ (H ≠> inv x) ∈ A by (metis inv-inv
    x(1))
    hence (H ≠> (x ⊗ x')) ◁▷ (H ≠> inv x) ∈ A by (metis rcos-sum x'(1)
    x(1))
    hence H ≠> (x ⊗ x' ⊗ inv x) ∈ A by (metis inv-closed m-closed
    rcos-sum x'(1) x(1))
    moreover have x ⊗ x' ⊗ inv x ∈ carrier G using x x' by (metis inv-closed
    m-closed)
    ultimately have x ⊗ x' ⊗ inv x ∈ {x ∈ carrier G. H ≠> x ∈ A} by auto
    hence xcoset : (x ⊗ x' ⊗ inv x) ⊗ x ∈ {x ∈ carrier G. H ≠> x ∈ A} ≠> x
    unfolding r-coset-def using invx by auto
    have (x ⊗ x' ⊗ inv x) ⊗ x = (x ⊗ x') ⊗ (inv x ⊗ x) by (metis Units-eq
    Units-inv-Units m-assoc m-closed x'(1) x(1))
    also have ... = x ⊗ x' using x(1) l-inv x'(1) m-closed r-one by auto
    also have ... = y by (metis (y = x ⊗ x'))
    finally have x ⊗ x' ⊗ inv x ⊗ x = y.
    with xcoset show y ∈ {x ∈ carrier G. H ≠> x ∈ A} ≠> x by auto
    qed
with assms show ?thesis by (metis (full-types) factgroup-subgroup-union-char
normal-imp-subgroup)
qed

lemma (in normal) factgroup-subgroup-union-factor:
  assumes subgroup A (G Mod H)
  shows A = rcosets G \Carrier := \bigcup A\} H
proof –
  have A = rcosets G \Carrier := {x ∈ carrier G. H ≠> x ∈ A}\} H
  proof auto
    fix U
    assume U:U ∈ A
    then obtain x' where x':x' ∈ carrier G U = H ≠> x' using assms sub-
    group.subset unfolding FactGroup-def RCOSETS-def by force
    with U have H ≠> x' ∈ A by simp
    with x' show U ∈ rcosets G \Carrier := {x ∈ carrier G. H ≠> x ∈ A}\} H un-
    folding RCOSETS-def r-coset-def by auto
  next
    fix U
    assume U:U ∈ rcosets G \Carrier := {x ∈ carrier G. H ≠> x ∈ A}\} H
    then obtain x' where x':x' ∈ {x ∈ carrier G. H ≠> x ∈ A} U = H ≠> x'
    unfolding RCOSETS-def r-coset-def by auto
    hence x' ∈ carrier G H ≠> x' ∈ A by auto
    with x' show U ∈ A by simp
    qed
with assms show ?thesis using factgroup-subgroup-union-char by auto
qed
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**

flattened \((G\colon\{\text{a set}, \, \text{b}\})\) monoid-scheme rep = (carrier= (rep \ (carrier G)),

\(\text{monoid.multip} = (\lambda x \, y. \, \text{rep} \ ((\text{the-inv-into} \, \text{carrier} \, G) \, \text{rep} \, x) \, \otimes_G \, \text{the-inv-into} \, \text{carrier} \, G \, \text{rep} \, y))\),

\(\text{one} = \text{rep} \, 1_G \) \)

**lemma flatten-set-group-hom:**

assumes group: group \(G\)

assumes inj-on rep (carrier G)

shows rep \(\in\) hom \(G\) (flatten \(G\) rep)

unfolding hom-def

proof auto

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

\(\text{assume} \, g\, \in\, \text{carrier} \, G\)

\(\text{thus} \, \text{rep} \, g \, \in \, \text{carrier} \, (\text{flatten} \, G \, \text{rep})\) unfolding flatten-def by auto

next

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

\(\text{assume} \, g\, \in\, \text{carrier} \, G \, \text{and} \, h\, \in\, \text{carrier} \, G\)

\(\text{hence} \, \text{rep} \, g \, \in \, \text{carrier} \, (\text{flatten} \, G \, \text{rep}) \, \text{rep} \, h \, \in \, \text{carrier} \, (\text{flatten} \, G \, \text{rep})\) unfolding flatten-def by auto

\(\text{hence} \, \text{rep} \, g \, \otimes_{\text{flatten} \, G \, \text{rep}} \, \text{rep} \, h\)

\(= \, \text{rep} \, (\text{the-inv-into} \, \text{carrier} \, G \, \text{rep} \, (\text{rep} \, g) \, \otimes_G \, \text{the-inv-into} \, \text{carrier} \, G \, \text{rep} \, (\text{rep} \, h))\) unfolding flatten-def by auto

\(\text{also have} \, \ldots \, = \, \text{rep} \, (g \, \otimes_G \, h) \, \text{using} \, \text{inj} \, g \, h \, \text{by} \, \text{metis} \, \text{the-inv-into-f-f}\)

\(\text{finally show} \, \text{rep} \, (g \, \otimes_G \, h) \, = \, \text{rep} \, g \, \otimes_{\text{flatten} \, G \, \text{rep}} \, \text{rep} \, h\).

qed

**lemma flatten-set-group:**

assumes group: group \(G\)

assumes inj-on rep (carrier G)

shows group (flatten \(G\) rep)

proof (rule groupI)

\(\text{fix} \, x \, y\)

\(\text{assume} \, x\, \cdot \, x \, \in \, \text{carrier} \, (\text{flatten} \, G \, \text{rep}) \, \text{and} \, y\, \cdot \, y \, \in \, \text{carrier} \, (\text{flatten} \, G \, \text{rep})\)

define \(g \, h\)

where \(g = \text{the-inv-into} \, \text{carrier} \, G \, \text{rep} \, x\)

and \(h = \text{the-inv-into} \, \text{carrier} \, G \, \text{rep} \, y\)

\(\text{hence} \, x \, \otimes_{\text{flatten} \, G \, \text{rep}} \, y \, = \, \text{rep} \, (g \, \otimes_G \, h)\) unfolding flatten-def by auto

moreover from \(g\,\text{-def} \, h\,\text{-def} \, \text{have} \, g \, \in \, \text{carrier} \, G \, h \, \in \, \text{carrier} \, G\)

using inj \(x \, \cdot \, y \, \text{the-inv-into-into}\) unfolding flatten-def by (metis partial-object.select-convs(1) subset-refl)+

\(\text{hence} \, g \, \otimes_G \, h \in \text{carrier} \, G \, \text{by} \, \text{metis} \, \text{group} \, \text{group.is-monoid} \, \text{monoid.m-closed}\)

\(\text{hence} \, \text{rep} \, (g \, \otimes_G \, h) \, \in \, \text{carrier} \, (\text{flatten} \, G \, \text{rep})\) unfolding flatten-def by simp

ultimately show \(x \, \otimes_{\text{flatten} \, G \, \text{rep}} \, y \, \in \, \text{carrier} \, (\text{flatten} \, G \, \text{rep})\) by simp

next
show 1 \( \text{flatten } G \text{ rep } \in \text{carrier } (\text{flatten } G \text{ rep}) \) unfolding flatten-def by (simp add: group group.is-monoid)

next

fix \( x \ y \ z \)

assume \( x \colon x \in \text{carrier } (\text{flatten } G \text{ rep}) \) and \( y \colon y \in \text{carrier } (\text{flatten } G \text{ rep}) \) and \( z \colon z \in \text{carrier } (\text{flatten } G \text{ rep}) \)

define \( g \ h \ k \)

where \( g = \text{the-inv-into } (\text{carrier } G) \text{ rep } x \)

and \( h = \text{the-inv-into } (\text{carrier } G) \text{ rep } y \)

and \( k = \text{the-inv-into } (\text{carrier } G) \text{ rep } z \)

hence \( x \circ \text{flatten } G \text{ rep } y \circ \text{flatten } G \text{ rep } z = (\text{rep } (g \otimes G \ h)) \otimes \text{flatten } G \text{ rep } z \)

unfolding flatten-def by auto

also have \( \ldots = \text{rep } (\text{the-inv-into } (\text{carrier } G) \text{ rep } (\text{rep } (g \otimes G \ h)) \otimes G \ k) \) using k-def unfolding flatten-def by auto

also from g-def h-def k-def have ghkG: \( g \in \text{carrier } G \ h \in \text{carrier } G \ k \in \text{carrier } G \)

using inj x y z the-inv-into unfolding flatten-def by fastforce+

hence ghkG \( g \circ G \ k \in \text{carrier } G \) and \( hk : h \circ G \ k \in \text{carrier } G \) by (metis group group.is-monoid monoid.m-closed)+

hence \( \text{rep } (\text{the-inv-into } (\text{carrier } G) \text{ rep } (\text{rep } (g \otimes G \ h)) \otimes G \ k) = \text{rep } ((g \otimes G \ h) \otimes G \ k) \)

unfolding flatten-def using inj the-inv-into ff by fastforce

also have \( \ldots = \text{rep } (g \circ G \ (h \circ G \ k)) \) using group group.is-monoid ghkG monoid.m-assoc by fastforce

also have \( \ldots = x \circ \text{flatten } G \text{ rep } (\text{rep } (h \circ G \ k)) \) unfolding g-def flatten-def using bk inj the-inv-into ff by fastforce

also have \( \ldots = x \circ \text{flatten } G \text{ rep } (y \circ \text{flatten } G \text{ rep } z) \) unfolding h-def k-def flatten-def using x y by force

finally show \( x \circ \text{flatten } G \text{ rep } y \circ \text{flatten } G \text{ rep } z = x \circ \text{flatten } G \text{ rep } (y \circ \text{flatten } G \text{ rep } z) \).

next

fix \( x \)

assume \( x \colon x \in \text{carrier } (\text{flatten } G \text{ rep}) \)

define \( g \) where \( g = \text{the-inv-into } (\text{carrier } G) \text{ rep } x \)

hence gG: \( g \in \text{carrier } G \) using inj x unfolding flatten-def using the-inv-into by force

have \( 1 \colon G \in (\text{carrier } G) \) by (simp add: group group.is-monoid)

hence the-inv-into (carrier G) rep (1 \( \text{flatten } G \text{ rep} \)) = 1 \( G \) unfolding flatten-def using the-inv-into ff inj by force

hence \( 1 \text{flatten } G \text{ rep } \otimes \text{flatten } G \text{ rep } x = \text{rep } (1 \text{G } \otimes G \ g) \) unfolding flatten-def g-def by simp

also have \( \ldots = \text{rep } g \) using gG group by (metis group.group.is-monoid monoid.l-one)

also have \( \ldots = x \) unfolding g-def using inj x f-the-inv-into ff 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 \( \in \text{hom } G \) (flatten G rep) using flatten-set-group-hom by auto

fix \( x \)
assume \( x : x \in \text{carrier} (\text{flatten } G \text{ rep}) \)

define \( g \) where \( g = \text{the-inv-into} (\text{carrier } G) \text{ rep } x \)

hence \( g : G \in \text{carrier } G \) using \( \text{inj } x \) unfolding \( \text{flatten-def} \) using \( \text{the-inv-into} \) by force

hence \( \text{invG} : \text{inv } G \in \text{carrier } G \) by \( \text{metis group group.inv-closed} \)

hence \( \text{rep} (\text{inv } G \text{ g}) \in \text{carrier} (\text{flatten } G \text{ rep}) \) unfolding \( \text{flatten-def} \) by auto

moreover have \( \text{rep} (\text{inv } G \text{ g}) \otimes \text{flatten } G \text{ rep } x = \text{rep} (\text{inv } G \text{ g}) \otimes \text{flatten } G \text{ rep } (\text{rep } g) \)

unfolding \( g \)-def using \( f \)-the-inv-into-f \( \text{inj } x \) unfolding \( \text{flatten-def} \) by fastforce

using \( \text{hom} \) unfolding \( \text{hom-def} \) using \( \text{gG} \text{ invG} \) \( \text{hom-def} \) by auto

hence \( \text{rep} (\text{inv } G \text{ g}) \otimes \text{flatten } G \text{ rep } x = 1 \text{ flatten } G \text{ rep } \) unfolding \( \text{flatten-def} \) by auto

ultimately show \( \exists y \in \text{carrier} (\text{flatten } G \text{ rep}). y \otimes \text{flatten } G \text{ rep } x = 1 \text{ flatten } G \text{ rep } \)

qed

lemma \( \text{(in normal)} \) \( \text{flatten-set-group-mod-inj} \):

shows \( \text{inj-on} (\lambda U. \text{SOME } g. g \in U) (\text{carrier } (G \text{ Mod } H)) \)

proof (rule \( \text{inj-onI} \))

fix \( U \) \( V \)

assume \( U : U \in \text{carrier } (G \text{ Mod } H) \) and \( V : V \in \text{carrier } (G \text{ Mod } H) \)

then obtain \( g \) \( h \) where \( g : U = H \not\rightarrow g \in \text{carrier } G \) and \( h : V = H \not\rightarrow h \in \text{carrier } G \)

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

hence notempty \( U \neq \{\} \neq \{\} \) 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 \( \text{metis IntI ex-in-conv somel} \)

thus \( U = V \) by \( \text{metis Int-iff g h is-subgroup repr-independence} \)

qed

lemma \( \text{(in normal)} \) \( \text{flatten-set-group-mod} \):

shows \( \text{group} (\text{flatten } (G \text{ Mod } H) (\lambda U. \text{SOME } g. g \in U)) \)

using \( \text{factorgroup-is-group} \) \( \text{flatten-set-group-mod-inj} \) by (rule \( \text{flatten-set-group} \))

lemma \( \text{(in normal)} \) \( \text{flatten-set-group-mod-iso} \):

shows \( (\lambda U. \text{SOME } g. g \in U) \in \text{iso} (G \text{ Mod } H) (\text{flatten } (G \text{ Mod } H) (\lambda U. \text{SOME } g. g \in U)) \)

unfolding \( \text{iso-def} \) \( \text{bij-beta-def} \)

apply (auto)

apply (metis \( \text{flatten-set-group-mod-inj} \) \( \text{factorgroup-is-group} \) \( \text{flatten-set-group-hom} \))

apply (rule \( \text{flatten-set-group-mod-inj} \))

unfolding \( \text{flatten-def} \) 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: \( \forall H. H \triangleleft G \implies (H = \text{carrier } G \lor H = \{1\}) \)

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 \( \neq \{1\} \) 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-nat-iff by auto
next
  fix H
  assume H \triangleleft 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 \lor card H = order G unfolding prime-nat-iff by simp
  thus H = \text{carrier } G \lor H = \{1\}
proof
  assume card H = 1
  moreover from HG have 1 \in 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 \subseteq \text{carrier } G unfolding subgroup-def by simp
  moreover from prime have card (carrier G) > 1 unfolding order-def prime-nat-iff ..
  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 \( H : \text{group} \ H \)

assumes \( \varphi \in \text{iso} \ G \ H \)

shows \( \text{simple-group} \ H \)

unfolding simple-group-def simple-group-axioms-def using assms(1)

proof (auto del: equalityI)

from iso have order \( G = \text{order} \ H \)

unfolding iso-def order-def using bij-betw-same-card

by auto

with order-gt-one show \( \text{Suc} \ 0 < \text{order} \ H \) by simp

next

have inv-iso \( \text{inv-into} \ (\text{carrier} \ G) \varphi \) \( \in \text{iso} \ G \ H \)

by (simp add: iso-set-sym)

fix \( N \)

assume \( NH : N < H \) and \( \text{Nneq1} : N \neq \{1\}_H \)

then interpret \( \text{Nnormal} : \text{normal} \ N \ H \) by simp

define \( M \) where \( M = (\text{inv-into} \ (\text{carrier} \ G) \varphi) \cdot N \)

hence \( \text{MG} : M < G \)

using inv-iso \( \text{NH} \ H \)

by (metis is-group iso-normal-subgroup)

have \( \text{surj} \cdot \varphi \cdot \text{carrier} \ G = \text{carrier} \ H \)

using iso unfolding iso-def bij-betw-def

by simp

hence \( \text{MN} : \varphi \cdot M = N \)

unfolding M-def using Nnormal subset image-inv-into-cancel

by metis

moreover have \( M \neq \{1\} \)

proof (rule notI)

assume \( M = \{1\} \)

hence \( \varphi \cdot M = \{1\} \)

by (metis (full-types) image-empty image-insert)

hence \( \varphi \cdot M = \{1\}_H \)

by (metis (lifting) Nnormal.is-subgroup MN calculation singleton-iff subgroup one-closed)

thus False using \( \text{Nneq1} \ MN \)

by simp

qed

hence \( M = \text{carrier} \ G \)

using no-real-normal-subgroup MG by auto

ultimately show \( N = \text{carrier} \ H \)

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:**

assumes \( \text{asm} : \text{simple-group} \ (G \ 	ext{Mod} \ (\text{carrier} \ G)) \)

proof

assume \( \text{asm:simple-group} \ (G \ 	ext{Mod} \ (\text{carrier} \ G)) \)

have \( \text{group} \ (G[\text{carrier} := \{1\}]) \)

by (metis subgroup-imp-group triv-subgroup)

with \( \text{asm self-factor-iso simple-group.iso-simple} \)

have simple-group \( (G[\text{carrier} := \{1\}]) \)

by auto

thus False using simple-group.simple-not-triv by force

qed

end

theory MaximalNormalSubgroups
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 = normal +
  assumes proper: $H \neq \text{carrier } G$
  assumes max-normal: $J \triangleleft G \Rightarrow J \neq H \Rightarrow J \neq \text{carrier } G \Rightarrow \neg (H \subseteq J)$

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

theorem (in normal) max-normal-simple-quotient:
  assumes finite: finite $(\text{carrier } G)$
  shows max-normal-subgroup $H G = \text{simple-group } (G \mod H)$

proof
  assume max-normal-subgroup $H G$
  then interpret $\max H$: max-normal-subgroup $H G$.
  show simple-group $(G \mod H)$ unfolding simple-group-def simple-group-axioms-def
  proof (intro conjI factgroup-is-group allI impI disjCI)
    from finite factgroup-finite factgroup-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 carrier $(G \mod H) \neq \{1 G \mod H\}$ using fact-group-trivial-iff by auto
    hence $1 \neq \text{order } (G \mod H)$ using factgroup-is-group group.order-one-triv-iff bymetis
    with $gt0$ show $1 < \text{order } (G \mod H)$ unfolding order-def FactGroup-def by auto
  next
    fix $A'$
    assume $A'normal: A' \triangleleft G \mod H$ and $A'nottriv: A' \neq \{1 G \mod H\}$
    define $A$ where $A = \bigcup A'$
    have $A2: A \triangleleft G$ using $A'normal$ unfolding $A$-def by (rule factgroup-subgroup-union-normal)
    have $H \in A'$ using $A'normal$ normal-imp-subgroup subgroup.one-closed unfolding FactGroup-def by force
    hence $H \subseteq A$ unfolding $A$-def by auto
    hence $A1: H \triangleleft (G \{\text{carrier } := A\})$ using $A2$ is-normal by (metis is-subgroup maxH.max-normal normal-restrict-supergroup subgroup-self)
    have $A3: A' = \text{rcosets } G \{\text{carrier } := A\} H$
    unfolding $A$-def using factgroup-subgroup-union-factor $A'normal$ normal-imp-subgroup by auto
    from $A1$ interpret normalHA: normal $H$ $(G \{\text{carrier } := A\})$ by metis
have \( H \subseteq A \) using \( \text{normal} \cdot \text{HA.is-subgroup} \cdot \text{subgroup} \cdot \text{subset} \) by force

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

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

proof

\begin{align*}
\text{assume } & A = H \\
\text{hence } & \text{carrier} \ (G[\text{carrier} := A] \ Mod \ H) = \{1(G[\text{carrier} := A] \ Mod \ H)\}
\end{align*}

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

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

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

with \( A \not\text{triv} \) show \( \text{thesis} \).

next

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

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

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

qed

qed

next

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

show \( \text{max-normal-subgroup} \ H \ G \)

proof

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

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

next

fix \( A \)

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

show \( \neg H \subseteq A \)

proof

assume \( HA: H \subseteq A \)

hence \( H < (G[\text{carrier} := A]) \) by (metis \( \text{A}(1) \) inv-op-closed \( \text{2.is-subgroup} \) normal inv op closed \( \text{normal-restrict-supergroup} \))

then interpret \( \text{normal} \cdot \text{HA: normal} \ H (G[\text{carrier} := A]) \) by simp

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

have \( \text{rcosets} (G[\text{carrier} := A]) H < G \ Mod \ H \) using normality-factorization

is-normal HA \( \text{A}(1) \) by auto

with simple have \( \text{rcosets} (G[\text{carrier} := A]) H = \{1_G \ Mod \ H\} \lor \text{rcosets} (G[\text{carrier} := A]) H = \text{carrier} \ (G \ Mod \ H) \)

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

thus \( \text{False} \)

proof

assume \( \text{rcosets} G[\text{carrier} := A] H = \{1_G \ Mod \ H\} \)

hence \( \text{rcosets} G[\text{carrier} := A] H = \{1_G[\text{carrier} := A] \ Mod \ H\} \) unfolding FactGroup-def by auto

with finite \( \text{A} \) have \( H = A \) using \( \text{normal} \cdot \text{HA.fact-group-trivial-iff} \) unfolding FactGroup-def by auto
with $A(2)$ show ?thesis by simp
next
  assume $A\triangleleft H;\text{rcosets}_{G\text{<carrier:=A}} H = \text{carrier } (G \Mod H)$
  have $A = \text{carrier } G$ unfolding FactGroup-def RCOSETS-def
proof
  show $A \subseteq \text{carrier } G$ using $A(1)$ normal-imp-subgroup subgroup.subset by metis
next
  show $\text{carrier } G \subseteq A$
proof
    fix $x$
    assume $x : x \in \text{carrier } G$
    hence $H \not\triangleright x \in \text{rcosets } H$ unfolding RCOSETS-def by auto
with $A\triangleleft H$ have $H \not\triangleright x \in \text{rcosets}_{G\text{<carrier:=A}} H$ unfolding FactGroup-def by simp
then obtain $x'$ where $x',x' \in A \triangleright H \not\triangleright x = H \not\triangleright G\text{<carrier:=A} \triangleright x'$ unfolding RCOSETS-def by auto
  hence $H \not\triangleright x \in A \triangleright x'$ unfolding r-coset-def by auto
  hence $x \in A \not\triangleright x'$ using $H A$ unfolding r-coset-def by auto
  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
  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 $\text{fin:finite } (\text{carrier } G)$
  assumes $\exists! Q. Q \in \text{subgroups-of-size } q$
  shows $(\text{THE } Q. Q \in \text{subgroups-of-size } q) < G$
proof
  from assms obtain Q where Q ∈ subgroups-of-size q by auto
  define Q where Q = (THE Q. Q ∈ subgroups-of-size q)
  with assms have Qsize:Q ∈ subgroups-of-size q using the1 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)
    unfolding conjugation-action-def by auto
    with invg g have invg <# (Q #> 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-nat-iff)
  define calM where calM = \{s. s ⊆ carrier G ∧ card s = q \^ 1\}
  define RelM where 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 snd-sylow-axioms-def snd-sylow-axioms-def
    using is-group primeq 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)
    then have card (subgroups-of-size q) = p ∨ card (subgroups-of-size q) = 1
      using primep by (auto simp add: prime-nat-iff)
    ultimately have card (subgroups-of-size q) = 1 using pq

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

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 \( \{ \text{THE } Q. \ Q \in \text{subgroups-of-size } q \} \triangleleft G \)
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\} \( \triangleleft G \)
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 \( \mathcal{S} \) 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 \( \mathcal{S} \)
assumes notempty:\( \mathcal{S} \neq [] \)
assumes hd:hd \( \mathcal{S} = \{1\} \)
assumes last:last \( \mathcal{S} = \text{carrier } G \)
assumes normal:\( \forall i. \ i + 1 < \text{length } \mathcal{S} \implies (\mathcal{S} ! i) \triangleleft G[\text{carrier := } \mathcal{S} ! (i + 1)] \)

**lemma (in normal-series)** is-normal-series: normal-series \( G \) \( \mathcal{S} \) 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 \( G [\{1\}, \text{carrier } G] \)
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 \( \mathcal{S} = 2 \)
shows \( \mathcal{S} = [\{1\}, \text{carrier } G] \)
proof(rule nth-equalityI)
from assms show length $\mathfrak{G} = \text{length} [\{1\}, \text{carrier} \ G]$ by auto

next

show $\mathfrak{G} \mathbin{!} i = [\{1\}, \text{carrier} \ G] \mathbin{!} i$ if $i: i < \text{length} \ \mathfrak{G}$ for $i$

proof -

have $i = 0 \lor i = 1$ using that assms by auto

thus $\mathfrak{G} \mathbin{!} i = [\{1\}, \text{carrier} \ G] \mathbin{!} i$ by auto

next

assume $i: i = 0$

with assms have $\mathfrak{G} \mathbin{!} i = \text{last} \ \mathfrak{G}$ by (metis diff-add-inverse last-conv-nth nat-1-add-1 notempty)

thus $\mathfrak{G} \mathbin{!} i = [\{1\}, \text{carrier} \ G] \mathbin{!} i$ using last by simp

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 $\text{normal} \ G (\mathfrak{G} \mid \text{carrier} := H)$
assumes $HG$

shows $\text{normal-series} \ G (\mathfrak{G} @ [\text{carrier} G])$

proof -

from normal interpret normalH: normal-series ($G \mid \text{carrier} := H$) $\mathfrak{F}$

from normalH.hd have $\text{hd} \mathfrak{F} = \{1\}$ by simp

with normalH.notempty have $\text{hdTriv:hd} (\mathfrak{F} @ [\text{carrier} G]) = \{1\}$ by (metis

hd-append2)

show $?\text{thesis}$ unfolding normal-series-def normal-series-axioms-def using is-group

proof auto

fix $x$

assume $x \in \text{hd} (\mathfrak{F} @ [\text{carrier} G])$

with $\text{hdTriv} \text{show} \ x = 1$ by simp

next

from $\text{hdTriv}$ show $1 \in \text{hd} (\mathfrak{F} @ [\text{carrier} G])$ by simp

next

fix $i$

assume $i: i < \text{length} \ \mathfrak{F}$

show $(\mathfrak{F} @ [\text{carrier} G]) \mathbin{!} i < G (\text{carrier} := (\mathfrak{F} @ [\text{carrier} G]) \mathbin{!} (i+1))$ by auto

proof (cases $i + 1 < \text{length} \ \mathfrak{F}$)

case True

with normalH.normal have $\mathfrak{F} \mathbin{!} i < G (\text{carrier} := \mathfrak{F} \mathbin{!} (i+1))$ by auto

with $i$ have $(\mathfrak{F} @ [\text{carrier} G]) \mathbin{!} i < G (\text{carrier} := \mathfrak{F} \mathbin{!} (i+1))$ using nth-append by metis

with True show $(\mathfrak{F} @ [\text{carrier} G]) \mathbin{!} i < G (\text{carrier} := (\mathfrak{F} @ [\text{carrier} G]) \mathbin{!} (Suc \ i))$ using nth-append Suc-eq-plus1 by metis

28
next case False
  with i have i2: i + 1 = length H by simp
from i have (H @ [carrier G]) ! i = H ! i by (metis nth-append)
also from i2 normalH notempty have ... = last H by (metis add-diff-cancel-right last-conv-nth)
also from normalH have ... = last H by simp
finally have (H @ [carrier G]) ! i = H by (metis nth-append-length)
ultimately show ?thesis using HG by auto
qed

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

lemma (in normal-series) normal-series-subgroups:
  shows $i < \text{length } G \Longrightarrow \text{subgroup } (G ! i) G$
proof (induction length G - (i + 2)
  case 0
    hence i: i + 2 = length G by simp
    hence ii: i + 1 = length G - 1 by force
  from i normal have $G ! i \triangleleft G (\{ \text{carrier } := G ! (i + 1) \}$ by auto
with ii last notempty show subgroup $(G ! i) G$ using last-conv-nth normal-imp-subgroup by fastforce
next case (Suc k)
  from Suc(3) normal have i: subgroup $(G ! i) (G (\text{carrier } := G ! (i + 1)))$
using normal-imp-subgroup by auto
  from Suc(2) have k: k = length G - ((i + 1) + 2) by arith
  with Suc have subgroup $(G ! (i + 1)) G$ by simp
  with i show subgroup $(G ! i) G$ by (metis is-group subgroup subgroup-of-subgroup)
qed

moreover have i + 1 = length G \Longrightarrow \text{subgroup } (G ! i) G
using last-conv-nth subgroup-of-subgroup by (metis add-diff-cancel-right' subgroup-self)
ultimately show $i < \text{length } G \Longrightarrow \text{subgroup } (G ! i) G$ by force
qed

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

lemma (in normal-series) normal-series-snd-to-last:
  shows $\text{length } G - 2 < G$
proof (cases 2 \leq \text{length } G)
case False
  with notempty have length: length G = 1 by (metis Suc-eq-plus1 le1 length-0-conv less-2-cases plus-nat.add-0)
  with hd have G ! (length G - 2) = \{ 1 \}$ using hd-conv-nth notempty by auto
  with length show ?thesis by (metis trivial-subgroup-is-normal)
next

  case True
  hence (length $\mathcal{G}$ - 2) + 1 < length $\mathcal{G}$ by arith
  with normal last have $\mathcal{G}$ ! (length $\mathcal{G}$ - 2) $\triangleleft$ G[(carrier := $\mathcal{G}$ ! ((length $\mathcal{G}$ - 2) + 1))] by auto
  have 1 + (1 + (length $\mathcal{G}$ - (1 + 1))) = length $\mathcal{G}$
    using True le-add-diff-inverse by presburger
  then have $\mathcal{G}$ ! (length $\mathcal{G}$ - 2) $\triangleleft$ G[(carrier := $\mathcal{G}$ ! (length $\mathcal{G}$ - 1))]
    by (metis $\mathcal{G}$ ! (length $\mathcal{G}$ - 2) $\triangleleft$ G [(carrier := $\mathcal{G}$ ! (length $\mathcal{G}$ - 2 + 1))],
        add.commute add-diff-cancel-left' one-add-one)
  with notempty last show ?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 $i \leq$ length $\mathcal{G}$ and $0 < i$
  shows normal-series (G[(carrier := $\mathcal{G}$ ! (i - 1))] (take i $\mathcal{G}$))
unfolding normal-series-def normal-series-axioms-def
using assms
apply (auto simp: hd del:equalityI)
  apply (simp add: is-group normal-series-subgroups subgroup.subgroup-is-group)
  apply (simp add: last-conv-nth min.absorb2 notempty)
using assms(1) normal apply simp
done

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:finite (carrier G)
  assumes orderG:order G = q * p
  assumes primep:prime p and primeq:prime q and pqq:prime $p < q$
  shows normal-series G [[1], (THE H. H $\in$ subgroups-of-size q), carrier G]
proof -
  define H where H = (THE H. H $\in$ subgroups-of-size q)
  with assms have HG:H $\triangleleft$ G by (metis pq-order-subgrp-normal)
  then interpret groupH: group G[(carrier := H)] unfolding normal-def by
  (metis subgroup-imp-group)
  have normal-series (G[(carrier := H)]) [[1], H] using groupH.trivial-normal-series
  by auto
  with HG show ?thesis unfolding H-def by (metis append-Cons append-Nil
    normal-series-extend)
qed

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

definition (in normal-series) quotients
  where quotients = map (\xi. G[(carrier := $\mathcal{G}$ ! (i + 1))]) Mod $\mathcal{G}$ ! i) [0..<((length $\mathcal{G}$) - 1)]
The list of quotient groups has one less entry than the series itself:

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

shows $\text{length quotients} + 1 = \text{length } G$

**proof** –

have $\text{length quotients} + 1 = \text{length } [0..<(\text{length } G - 1)] + 1$ unfolding quotients-def by simp

also have $\ldots = (\text{length } G - 1) + 1$ by (metis zero length-

also with notempty 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 $\text{last quotients } = G \mod G ! (\text{length } G - 1 - 1)$

**proof** –

from assms have $\text{lsimp: length } G - 1 - 1 = \text{length } G - 1$ by auto

from assms have $\text{quotients } \neq \text{quotients-def}$ by auto

hence $\text{last quotients } = \text{quotients ! (length quotients - 1)}$ by (metis last-conv-nth)

also have $\ldots = \text{quotients ! (length } G - 1 - 1)$ by (metis add-diff-cancel-left' quotients-length add . commute)

also have $\ldots = G ![\text{carrier := } G ! ((\text{length } G - 1 - 1)) \mod G ! (\text{length } G - 1 - 1)]$

unfolding quotients-def using assms by auto

also have $\ldots = G ![\text{carrier := } G ! (\text{length } G - 1)] \mod G ! (\text{length } G - 1 - 1)$ using lsimp by simp

also have $\ldots = G \mod G ! (\text{length } G - 1 - 1)$ 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 $\text{iso} : \Psi \in \text{iso } G H$

shows $\text{normal-series } H ![\text{map (image } \Psi ) G ]$

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

using $H \text{ notempty apply simp}$

**proof (rule conjI)**

from $H \text{ is-group iso}$ have $\text{group-hom: group-hom } G H \Psi$ unfolding group-hom-def group-hom-axioms-def iso-def by auto

have $\text{hd (map (image } \Psi ) G ) = \Psi ![\{1\}]$ by (metis hd-map 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 $\text{hd (map ('' ) } G ) = \{1_H\}$.

**next**

show $\text{last (map ('' ) } G ) = \text{carrier } H \land (\forall i. \text{Suc } i < \text{length } G \longrightarrow \Psi ![\text{carrier := } \Psi ![\text{carrier := } G ! \text{Suc } i]]$
proof (auto del: equalityI)

have last (map (('a) Ψ) G) = Ψ (carrier G) using last last-map notempty by metis
also have \dots = carrier H using iso unfolding iso-def bij_betw_def by simp
finally show last (map (('a) Ψ) G) = carrier H.

next
fix i
assume i: Suc i < length G
hence norm:G ! i < G[(carrier := G ! Suc i)] using normal by simp
moreover have restrict Ψ (G ! Suc i) ∈ iso (G[carrier := G ! Suc i])
(H[carrier := Ψ (G ! Suc i)])
by (metis H i is-group iso iso_restrict normal-series-subgroups)
moreover have group (G[carrier := G ! Suc i]) by (metis i normal-series-subgroups subgroup_imp_group)
moreover hence subgroup (Ψ ! Suc i) G by (metis i normal-series-subgroups subgroup_imp_group)
hence subgroup (Ψ ! Suc i) H by (metis H is-group iso iso-subgroup)
hence group (H[carrier := Ψ ! Suc i]) by (metis H subgroup subgroup_is_group)
ultimately have restrict Ψ (G ! Suc i) ! i < H[carrier := Ψ ! G ! Suc i]
using is-group H iso-normal-subgroup by (auto cong del: image_cong simp)
moreover from norm have G ! i ⊆ G ! Suc i unfolding normal_def subgroup_def
by auto
hence \{ y. ∃ x ∈ G ! i. y = (if x ∈ G ! i then Ψ x else undefined)\} = \{ y. ∃ x ∈ G ! i. y = Ψ x \} by auto
ultimately show Ψ ! Suc i ! i < H[carrier := Ψ ! G ! Suc i] unfolding restrict_def image_def by auto
qed
qed

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: \( \And i. i + 1 < \text{length } G \Rightarrow \text{simple-group } (G[\text{carrier := G} ! (i + 1)] \mod G ! i) \)

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 (\text{length } G = 1) = (G = \{1\})
proof
assume length G = 1
with hd have length G = length \{1\} \land (∀ i < length G. G ! i = \{1\} ! i)
using hd_conv_nth notempty by force

32
thus $G = [\{1\}]$ using list-eq iff nth-eq by blast

next
assume $G = \{1\}$
thus $\text{length } G = 1$ by simp

qed

lemma (in composition-series) composition-series-triv-group:
shows $(\text{carrier } G = \{1\}) = (G = ([1])$

proof
assume $G:\text{carrier } G = \{1\}$

have $\text{length } G = 1$
proof (rule ccontr)

assume $\text{length } G \neq 1$
with notempty have $\text{length } length G \geq 2$ 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 $(G(\text{carrier } := G ! \{1\}) \text{ by force}$
moreover have $SG:\text{subgroup } (G ! \{1\}) G$ using length normal-series-subgroups
by auto

hence $\text{group } (G(\text{carrier } := G ! \{1\}) \text{ by (metis subgroup-imp-group}$
ultimately have simple-group $(G(\text{carrier } := G ! \{1\}) \text{ using group.trivial-factor-iso}$

moreover from $SG G$ have carrier $(G(\text{carrier } := G ! \{1\}) = \{1\}$ unfolding subgroup-def by auto

ultimately show False using simple-group.simple-not-triv by force

thus $G = \{1\}$ by (metis composition-series-length-one)

next
assume $G = \{1\}$
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 $i + 1 < \text{length } G$
assumes $i > 0$
shows $G ! i \neq \{1\}$

proof
from assms have $(i - 1) + 1 < \text{length } G$ by simp

hence simple:simple-group $(G(\text{carrier } := G ! ((i - 1) + 1)) \text{ Mod } G ! (i - 1))$

using simplefact by auto

assume $i:G ! i = \{1\}$
moreover from assms have $(i - 1) + 1 = i$ by auto
ultimately have $G(\text{carrier } := G ! ((i - 1) + 1)) \text{ Mod } G ! (i - 1) = G(\text{carrier } := \{1\}) \text{ Mod } G ! (i - 1)$ using $i$ by auto

hence order $(G(\text{carrier } := G ! ((i - 1) + 1)) \text{ Mod } G ! (i - 1) = 1$ unfolding FactGroup-def order-def RCOSETS-def by force

thus False using $i$ simple unfolding simple-group-def simple-group-axioms-def
A composition series of a simple group always is its trivial one.

**Lemma** (in composition-series) composition-series-simple-group:
shows \((\text{simple-group } G) = (\mathcal{G} = \{\{1\}, \text{carrier } G\})\)

**Proof**
assume \(G = \{1\}, \text{carrier } G\]
with simplefact have simple-group \((G \mod \{1\})\) by auto
moreover have the-elem \(\in \text{iso} (G \mod \{1\}) G\) by (rule trivial-factor-iso)
ultimately show simple-group \(G\) by (metis is-group simple-group.iso-simple)

next
assume simple:
have length \(\mathcal{G} > 1\)
proof (rule ccontr)
assume \(\neg \, 1 < \text{length } \mathcal{G}\)
hence \(\text{length } \mathcal{G} = 1\) by (simp add: Suc-leI antisym notempty)
hence \(\text{carrier } G = \{1\}\) using hd last by (metis composition-series-length-one composition-series-triv-group)
hence \(\text{order } G = 1\) unfolding order-def by auto
with simple show False unfolding simple-group-def simple-group-axioms-def by auto
qed
moreover have \(\text{length } \mathcal{G} \leq 2\)
proof (rule ccontr)
define \(k\) where \(k = \text{length } \mathcal{G} - 2\)
assume \(\neg \, (\text{length } \mathcal{G} \leq 2)\)
hence \(gt2: \text{length } \mathcal{G} > 2\) by simp
hence \(ksmall:k + 1 < \text{length } \mathcal{G}\) unfolding k-def by auto
from \(gt2\) have \(\text{carrier:} \mathcal{G} ! (k + 1) = \text{carrier } G\) unfolding notempty last last-conv-nth k-def
by (metis Nat.add-diff-assoc Nat.diff-cancel \(\neg \, \text{length } \mathcal{G} \leq 2\) add.commute Nat.le-linear one-add-one)
from normal ksmall have \(\mathcal{G} ! k < G\ (\text{carrier := } \mathcal{G} ! (k + 1))\) by simp
from simplefact ksmall have simplek:simple-group \((G\ (\text{carrier := } \mathcal{G} ! (k + 1)))\)
Mod \(\mathcal{G} ! k\) by simp
from simplefact ksmall have simplek:simple-group \((G\ (\text{carrier := } \mathcal{G} ! ((k - 1) + 1)))\)
Mod \(\mathcal{G} ! (k - 1))\) by auto
have \(\mathcal{G} ! k < G\) using carrier k-def gt2 normal ksmall by force
with simple have \((\mathcal{G} ! k) = \text{carrier } G \lor (\mathcal{G} ! k) = \{1\}\) unfolding simple-group-def simple-group-axioms-def by simp
thus False
proof (rule disjE)
assume \(\mathcal{G} ! k = \text{carrier } G\)
hence \(G\ (\text{carrier := } \mathcal{G} ! (k + 1))\) Mod \(\mathcal{G} ! k = G\ Mod (\text{carrier } G)\) using carrier by auto
with simplek self-factor-not-simple show False by auto
next
assume \(\mathcal{G} ! k = \{1\}\)
with ksmall k-def gt2 show False using inner-elements-not-triv by auto
qed
qed
ultimately have length ℰ = 2 by simp
thus ℰ = \{\{1\}, carrier G\} by (rule length-two-unique)
qed

Two consecutive elements in a composition series are distinct.

lemma (in composition-series) entries-distinct:
  assumes finite:finite (carrier G)
  assumes i:i + 1 < length ℰ
  shows ℰ ! i ≠ ℰ ! (i + 1)
proof
  from finite have finite (ℰ ! (i + 1))
    using i normal-series-subgroups subgroup.subset rev-finite-subset by metis
  hence fin:finite (carrier (G[carrier := ℰ ! (i + 1)])) by auto
  from i have norm:ℰ ! i ⊂ (G[carrier := ℰ ! (i + 1)]) by (rule normal)
  assume ℰ ! i = ℰ ! (i + 1)
  hence ℰ ! i = carrier (G[carrier := ℰ ! (i + 1)]) by auto
  hence carrier ((G[carrier := (ℰ ! (i + 1))]) Mod (ℰ ! i)) = \{1(G[carrier := ℰ ! (i + 1)]) Mod ℰ ! i\}
    using norm fin normal-fact-group-trivial-iff by metis
  hence ¬ simple-group ((G[carrier := (ℰ ! (i + 1))]) Mod (ℰ ! i)) by (metis simple-group.simple-not-triv)
  thus False by (metis i simplefact)
qed

The normal series for groups of order \(p \cdot q\) is even a composition series:

lemma (in group) pq-order-composition-series:
  assumes finite:finite (carrier G)
  assumes orderG:order G = q * p
  assumes primep:prime p and primeq:prime q and pq:p < q
  shows composition-series G \{\{1\}, \{THE H. H ∈ subgroups-of-size q\}, carrier G\}
unfolding composition-series-def composition-series-axioms-def
apply(auto)
using assms apply(rule pq-order-normal-series)
proof
  define H where H = (THE H. H ∈ subgroups-of-size q)
  from assms have exi:∃!Q. Q ∈ (subgroups-of-size q) by (auto simp: pq-order-unique-subgrp)
    hence Hsize:H ∈ subgroups-of-size q unfolding H-def using theI' by metis
  hence HSubG:subgroup H G unfolding subgroups-of-size-def by auto
  then interpret Hgroup: group G[carrier := H] by (metis subgroup-imp-group)
  fix i
  assume i < Suc (Suc 0)
  hence i = 0 ∨ i = 1 by auto
  thus simple-group (G[carrier := [H, carrier G] ! i]) Mod \{\{1\}, H, carrier G\} ! i)
    proof
      assume i:i = 0

35
from Hsize have orderH:order (G⟨carrier := H⟩) = q unfolding subgroups-of-size-def
order-def by simp
  hence order (G⟨carrier := H⟩ Mod {1}) = q unfolding FactGroup-def using
card-rcosets-triv order-def
    by (metis Hgroup.card-rcosets-triv HsubG finite monoid.cases-scheme monoid.select-cones(2)
      partial-object.select-cones(1) partial-object.update-convs(1) subgroup-finite)
  have normal {1} (G⟨carrier := H⟩) by (metis Hgroup.is-group Hgroup.normal-inv-iff
HsubG group.trivial-subgroup-is-normal is-group singleton-iff subgroup.one-closed
subgroup.subgroup-of-subgroup)
  hence group (G⟨carrier := H⟩ Mod {1}) by (metis normal_factorgroup-is-group)
    with orderH primeq have simple-group (G⟨carrier := H⟩ Mod {1}) by (metis
order ⟨G⟨carrier := H⟩ Mod {1}⟩ = q group_prime-order-simple)
    with i show thesis by simp
next
  assume i:i = 1
from assms exi have H < G unfolding H-def by (metis pq-order-subgrp-normal)
  hence groupGH:group (G Mod H) by (metis normal_factorgroup-is-group)
  from primeq have q ≠ 0 by (metis not-prime-0)
  from HsubG finite orderG have card (rcosets H) * card H = q * p unfolding
subgroups-of-size-def using lagrange by simp
    with Hsize have card (rcosets H) * q = q * p unfolding subgroups-of-size-def
by simp
      with (q ≠ 0) have card (rcosets H) = p by auto
    hence order (G Mod H) = p unfolding ordereq FactGroup-def by auto
      with groupGH primeq have simple-group (G Mod H) by (metis group_prime-order-simple)
        with i show thesis by auto
qed

Prefixes of composition series are also composition series.

lemma (in composition-series) composition-series-prefix-closed:
  assumes i ≤ length G and 0 < i
  shows composition-series (G⟨carrier := G ! (i - Suc 0)⟩) (take i G)
unfolding composition-series-def composition-series-axioms-def
proof auto
  from assms show normal-series (G⟨carrier := G ! (i - 1)⟩) (take i G) by
    (metis One-nat-def normal-series-prefix-closed)
next
  fix j
  assume j:Suc j < length G Suc j < i
    with simplefact show simple-group (G⟨carrier := G ! Suc j⟩ Mod G ! 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 ≤ length G
  shows simple-group (G⟨carrier := G ! 1⟩)
proof –
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 : nat < length G`

**shows** `(simple-group (G (carrier := G ! i))) = (i = 1)`

**Proof**

**Assume** `simp : simple-group (G (carrier := G ! i))`

**Hence** `G ! i ≠ {1}` using `simple-group.simple-not-triv` by `force`

**Hence** `i ≠ 0` using `hd hd-cons nth notempty` by `auto`

**Then interpret** `compTake : composition-series G (carrier := G ! i)` take `(Suc i) G`

**Using** `assms composition-series-prefix-closed` by `(metis diff-Suc-1 less-eq-Suc-le zero-less-Suc)`

**From** `simp` have `(take (Suc i) G) = [{1} G (carrier := G ! i)]`, carrier `(G (carrier := G ! i))`

**By** `(metis compTake.composition-series-simple-group)`

**Hence** `length (take (Suc i) G) = 2` by `auto`

**Hence** `min (length G) (Suc i) = 2` by `(metis length-take)`

**With** `assms` have `Suc i = 2` by `force`

**Thus** `i = 1` by `simp`

**Next**

**Assume** `i : i = 1`

**With** `assms` have `2 ≤ length G` by `simp`

**With** `i` show `simple-group (G (carrier := G ! 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 (carrier G)`

**Assumes** `length G > 1`

**Shows** `max-normal-subgroup (G ! (length G - 2)) G`

**Unfolding** `max-normal-subgroup-def max-normal-subgroup-axioms-def`

**Proof** (auto del: equalityI)

**Show** `G ! (length G - 2) < G` by `(rule normal-series-snd-to-last)`

**Next**

**Define** `G'` where `G' = G ! (length G - 2)`

**From** `length G` have `length21 : length G - 2 + 1 = length G - 1` by `arith`
from length have length $\emptyset - 2 + 1 < length \emptyset$ by arith
with simplefact have simple-group $(G[\text{carrier} := \emptyset ! ((length \emptyset - 2) + 1)])$
Mod $G'$ unfolding G'-def by auto
with length±1 have simple-last:simple-group $(G \text{ Mod } G')$ using last notempty
last-conv-nth by fastforce

{ assume snd-to-last-eq:G' = carrier G
  hence carrier $(G \text{ Mod } G') = \{1 \text{ Mod } G'\}$
  using normal-series-snd-to-last finite normal.fact-group-trivial-iff unfolding
G'-def by metis
  with snd-to-last-eq have ¬ simple-group $(G \text{ Mod } G')$ 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 J \neq G' J \neq carrier G G' \subseteq J
  hence JG'GG':rcosets $(G[\text{carrier} := J])$ G' < G \text{ Mod } G' using normality-factorization
  normal-series-snd-to-last unfolding G'-def by auto
  from G'G J(1,4) have G'JG' < (G[\text{carrier} := J]) by (metis normal-imp-subgroup
  normal-imp-subgroup normal-imp-subgroup)
  from finite J(1) have finJ:finite J by (auto simp: normal-imp-subgroup
  subgroup-finite)
    from JG'GG' simple-last have rcosets $(G[\text{carrier} := J])$ G' = \{1 \text{ Mod } G'\} ∨
    rcosets $(G[\text{carrier} := J])$ G' = carrier $(G \text{ Mod } G')$
    unfolding simple-group-def simple-group-axioms-def by auto
  thus False
  proof
    assume rcosets $(G[\text{carrier} := J])$ G' = \{1 \text{ Mod } G'\}
    hence rcosets $(G[\text{carrier} := J])$ G' = \{1(G[\text{carrier} := J]) \text{ Mod } G'\} unfolding
    FactGroup-def by simp
    hence G' = J using G'J finJ normal.fact-group-trivial-iff unfolding
    FactGroup-def by fastforce
    with J(2) show False by simp
  next
    assume facts-eq:rcosets $(G[\text{carrier} := J])$ G' = carrier $(G \text{ Mod } G')$
    have J = carrier G
    proof
      show J \subseteq carrier G using J(1) normal-imp-subgroup subgroup subset by
      force
    next
      show carrier G \subseteq J
      proof
        fix x
        assume x:x \in carrier G
        hence G' #> x \in carrier $(G \text{ Mod } G')$ unfolding FactGroup-def
        RCOSETS8-def by auto
        hence G' #> x \in rcosets $(G[\text{carrier} := J])$ G' using facts-eq by auto
  }
then obtain \( j \) where \( j : G' \# > x = G' \# > j \) unfolding RCOSETS-def r-coset-def by force

hence \( x \in G' \# > j \) using G'G normal-imp-subgroup x repr-independenceD by fastforce

then obtain \( g' \) where \( g' : g' \in G' \times = g' \otimes j \) unfolding r-coset-def by auto

hence \( g' \in J \) using G'J normal-imp-subgroup subgroup subset by force

with \( g'(j) \) show \( x \in J(1) \) using J(1) normal-imp-subgroup subgroup m-closed by fastforce

qed

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

\textbf{Lemma} remdups-adj-obtain-adjacency:
\textbf{Assumes} \( i + 1 < \text{length} (\text{remdups-adj} \ x \ s) \text{ length} \ x \ s > 0 \)
\textbf{Obtains} \( j \) where \( j + 1 < \text{length} \ x \ s \)

\( (\text{remdups-adj} \ x \ s) ! i = x ! j \ (\text{remdups-adj} \ x \ s) ! (i + 1) = x ! (j + 1) \)

\textbf{Using} \textbf{Proof} (induction \( x \ s \) arbitrary: \( i \) thesis)

\textbf{Case} Nil

\textbf{Hence} \( \text{False} \) by (metis length-greater-0-conv)

\textbf{Thus} thesis.

\textbf{Next}

\textbf{Case} (\( \text{Cons} \ x \ s \))

\textbf{Then have} \( x \ s \neq [] \)

by auto

\textbf{Then obtain} \( y \ x \ s' \) where \( x : x = y \# x \ s' \)

by (cases \( x \ s \)) blast

from \( x \ s \neq [] \) have \( \text{lenxs:length} \ x \ s > 0 \) by simp

from \( x \ s \) have \( \text{rem:remdups-adj} \ (x \# x \ s) = (\text{if} x = y \text{ then remdups-adj} \ (y \# x \ s') \text{ else} x \# \text{remdups-adj} \ (y \# x \ s')) \) using \( \text{remdups-adj.simps}(3) \) by auto

\textbf{Show} thesis

\textbf{Proof} (cases \( x = y \))

\textbf{Case} True

with \( \text{rem} \ x \ s \) have \( \text{rem2:remdups-adj} \ (x \# x \ s) = \text{remdups-adj} \ x \ s \) by auto

with \( \text{Cons}(3) \) have \( i + 1 < \text{length} (\text{remdups-adj} \ x \ s) \) by simp

with \( \text{Cons} \IH \text{lenxs obtain} \ k \) where \( j : k + 1 < \text{length} \ x \ s \ \text{remdups-adj} \ x \ s ! i \)

= \( x \ s ! k \)

\( \text{remdups-adj} \ x \ s ! (i + 1) = x \ s ! (k + 1) \) by auto

\textbf{Thus} thesis using \( \text{Cons}(2) \) rem2 by auto

\textbf{Next}

\textbf{Case} False

with \( \text{rem} \ x \ s \) have \( \text{rem2:remdups-adj} \ (x \# x \ s) = x \# \text{remdups-adj} \ x \ s \) by auto

\textbf{Show} thesis

\textbf{Proof} (cases \( i \))

\textbf{Case} 0
have $0 + 1 < \text{length} (x \# xs)$ using \text{lenxs} by auto
moreover have \text{remdups-adj} (x \# xs) ! i = (x \# xs) ! 0
proof –
  have \text{remdups-adj} (x \# xs) ! i = (x \# \text{remdups-adj} (y \# xs')) ! 0 using
  \text{xs} \text{rem2} 0 \text{ by simp}
  also have \ldots = x \text{ by simp}
  also have \ldots = (x \# xs) ! 0 \text{ 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} \text{rem2} 0 \text{ by simp}
  also have \ldots = \text{remdups-adj} (y \# xs') ! 0 \text{ by simp}
  also have \ldots = (y \# \text{remdups-adj} (y \# xs')) ! 0 \text{ by (metis nth-Cons')}
  \text{remdups-adj-Cons-alt}
  also have \ldots = y \text{ by simp}
  also have \ldots = (x \# xs) ! (0 + 1) \text{ unfolding xs by simp}
  finally show \text{thesis}.
qed
ultimately show \text{thesis} by (rule Cons.prems(1))
next
case (Suc $k$)
with \text{Cons(3)} have $k + 1 < \text{length} (\text{remdups-adj} (x \# xs)) - 1$ by auto
also have \ldots $\leq \text{length} (\text{remdups-adj} xs) + 1 - 1$ by (metis One-nat-def
  \text{le-refl list.size(4) rem2})
also have \ldots $= \text{length} (\text{remdups-adj} xs)$ by simp
finally have $k + 1 < \text{length} (\text{remdups-adj} xs)$.
with \text{Cons.IH lenxs} obtain $j$ where $j; j + 1 < \text{length} xs \text{remdups-adj} xs ! k
= \text{xs} ! j$
  \text{remdups-adj} xs ! (k + 1) = xs ! (j + 1) \text{ by auto}
from $j(1)$ have Suc $j + 1 < \text{length} (x \# xs)$ by simp
moreover have \text{remdups-adj} (x \# xs) ! i = (x \# xs) ! (Suc $j$)
proof –
  have \text{remdups-adj} (x \# xs) ! i = (x \# \text{remdups-adj} xs) ! i using \text{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 \text{thesis} .
qed
moreover have \text{remdups-adj} (x \# xs) ! (i + 1) = (x \# xs) ! (Suc $j + 1$
proof –
  have \text{remdups-adj} (x \# xs) ! (i + 1) = (x \# \text{remdups-adj} xs) ! (i + 1)
using \text{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 \text{thesis}.
ultimately show thesis by (rule Cons.prems(1))

lemma hd-remdups-adj[simp]: hd (remdups-adj xs) = hd xs
  by (induction xs rule: remdups-adj.induct) simp-all

lemma remdups-adj-adjacent:
  Suc i < length (remdups-adj xs) ⟹ remdups-adj xs ! i ≠ 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 ◁ G
  shows composition-series (G[carrier := K]) (remdups-adj (map (λH. K ∩ H) G))
unfolding composition-series-def composition-series-axioms-def normal-series-def
normal-series-axioms-def
apply (auto simp only: conjI del: equalityI)
proof –
  show group (G[carrier := K]) 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 ((∩) K) G) = []
  hence map ((∩) K) G = [] by (metis remdups-adj-Nil-iff)
  hence 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 G = {1} # tl G using notempty hd by (metis list.sel(1,3) neq-Nil-conv)
  hence map ((∩) K) G = map ((∩) K) ({1} # tl G) by simp
  hence remdups-adj (map ((∩) K) G) = remdups-adj ((K ∩ {1}) # (map ((∩) K) (tl G))) by simp
  also have ... = (K ∩ {1}) # tl (remdups-adj ((K ∩ {1}) # (map ((∩) K) (tl G)))) by simp
  finally have hd (remdups-adj (map ((∩) K) G)) = K ∩ {1} using list.sel(1)
  by metis
  thus hd (remdups-adj (map ((∩) K) G)) = {1}(carrier := K)
  using KG normal-imp-subgroup subgroup-one-closed by force
next
  — Show that the last entry is really $K ∩ G$. Since we don’t have a lemma ready
to talk about the last entry of a reduced list, we reverse the list twice.

**have** \( \text{rev } G = \langle \text{carrier } G \rangle \neq \text{tl } (\text{rev } G) \) by (metis list.sel(1,3) last-rev neq-Nil-conv notempty rev-is-Nil-conv rev-rev-ident)

**hence** \( \text{rev:rev } (\text{map } ((\cap ) \text{ } K) \text{ } G) = \langle K \cap (\text{carrier } G) \rangle \neq \text{tl } (\text{rev } G) \) by (metis rev-map)

**hence** \( \text{rev:rev } (\text{map } ((\cap ) \text{ } K) \text{ } G) = (K \cap (\text{carrier } G)) \neq (\text{map } ((\cap ) \text{ } K) \text{ } (\text{tl } (\text{rev } G))) \) by simp

**have** last (\text{remdups-adj } (\text{map } ((\cap ) \text{ } K) \text{ } G)) = \text{hd } (\text{rev } (\text{remdups-adj } (\text{map } ((\cap ) \text{ } K) \text{ } G)))

by (metis \text{hd-rev map-is-Nil-conv notempty rev-rev-ident})

**also have** \( \ldots = \text{hd } (\text{remdups-adj } (\text{rev } (\text{map } ((\cap ) \text{ } K) \text{ } G))) \) by (metis \text{remdups-adj-rev})

**also have** \( \ldots = \text{hd } (\text{remdups-adj } ((K \cap (\text{carrier } G)) \neq (\text{map } ((\cap ) \text{ } K) \text{ } (\text{tl } (\text{rev } G))))))) \) by (metis \text{list.sel}(1) \text{remdups-adj-Cons-alt})

**also have** \( \ldots = \text{K using } \text{KG normal-imp-subgroup subgroup.subset by force}

finally show last (\text{remdups-adj } (\text{map } ((\cap ) \text{ } K) \text{ } G)) = \text{carrier } (G[[\text{carrier } := K]]) \)

by auto

next

— The induction step, using the second isomorphism theorem for groups.

**fix** \( j \)

**assume** \( j; j + 1 < \text{length } (\text{remdups-adj } (\text{map } ((\cap ) \text{ } K) \text{ } G)) \)

**have** \( \text{KGnotempty:} (\text{map } ((\cap ) \text{ } K) \text{ } G) \neq [\] using notempty by (metis \text{Nil-is-map-conv})

**with** \( j \) **obtain** \( i \) **where** \( i: i + 1 < \text{length } (\text{map } ((\cap ) \text{ } K) \text{ } G) \)

(\text{remdups-adj } (\text{map } ((\cap ) \text{ } K) \text{ } G)) ! j = (\text{map } ((\cap ) \text{ } K) \text{ } G) ! i

(\text{remdups-adj } (\text{map } ((\cap ) \text{ } K) \text{ } G)) ! (j + 1) = (\text{map } ((\cap ) \text{ } K) \text{ } G) ! (i + 1)

using \( \text{remdups-adj-obtain-adjacency by force}

from \( i(1) \) **have** \( i^\prime; i + 1 < \text{length } G \) by (metis \text{length-map})

**hence** \( \text{GiSi}: G \cap i < G[[\text{carrier } := G ! (i + 1)]] \) by (metis \text{normal})

**hence** \( \text{GiSi}^\prime; G ! i \subseteq G ! (i + 1) \) using \( \text{normal-imp-subgroup subgroup.subset by force}

from \( i^\prime \) **have** \( \text{GiSi:finite } (G ! (i + 1)) \) using \( \text{normal-series-subgroups finite}

by (metis \text{subgroup-finite})

from \( \text{GiSi KG i' normal-series-subgroups have } \text{GiSiKnormGSi: G} ![i + 1] \cap K \triangleleft G[[\text{carrier } := G ! (i + 1)]] \)

using \( \text{second-isomorphism-grp.normal-subgrp-intersection-normal}

unfolding \( \text{second-isomorphism-grp-def second-isomorphism-grp-axioms-def by auto}

with \( \text{GiSi have } G ! i \cap (G ! (i + 1) \cap K) \triangleleft G[[\text{carrier } := G ! (i + 1)]] \)

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

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

**hence** \( \text{KKnormGSi:K} \cap G ! i \triangleleft G[[\text{carrier } := G ! (i + 1)]] \) using \( \text{GiSi' by auto}

**moreover have** \( K \cap G ! i \subseteq K \cap G ! (i + 1) \) using \( \text{GiSi' by auto}

**moreover have** \( \text{groupGSi:group } (G[[\text{carrier } := G ! (i + 1)]]) \) using \( i \) \text{normal-series-subgroups subgroup-imp-group by auto}

**moreover have** \( \text{subKGSiGSi:subgroup } (K \cap G ! (i + 1)) (G[[\text{carrier } := G ! (i + 1)]]) \) (G[[carrier := G ! (i + 1)]]) \) using \( i \text{normal-series-subgroups subgroup-imp-group by auto}


ultimately have fstgoal: K ∩ G ! i < G\{carrier := G ! (i + 1), carrier := K ∩ G ! (i + 1)}

using group.normal-restrict-supergroup by force
thus remdups-adj (map ((∩) K) G) ! j < G\{carrier := K, carrier := remdups-adj (map ((∩) K) G) ! (j + 1)}

using i by auto
from simplefact have Gisimple:simple-group (G\{carrier := G ! (i + 1)} Mod G ! i) using i \i by simp
hence Gimax:max-normal-subgroup (G ! i) (G\{carrier := G ! (i + 1)}
using normal.max-normal-simple-quotient GiSi finGSi by force
from GSiKnormGSi GSi have G ! i <#> G\{carrier := G ! (i + 1)} G ! (i + 1)
∩ K < (G\{carrier := G ! (i + 1)})
using groupGSi group.normal-subgroup-set-mult-closed set-mult-consistent by fastforce
hence G ! i <#> G ! (i + 1) ∩ K < G\{carrier := G ! (i + 1)} unfolding set-mult-def by auto
hence G ! i <#> K ∩ G ! (i + 1) < G\{carrier := G ! (i + 1)} using inf-commute by metis
moreover have G ! i ⊆ G ! i <#> G\{carrier := G ! (i + 1)} K ∩ G ! (i + 1)
unfolding second-isomorphism-grp-def second-isomorphism-grp-axioms-def
using subKGSiGSi GiSi normal-imp-subgroup by fastforce
hence G ! i ⊆ G ! i <#> K ∩ G ! (i + 1) unfolding set-mult-def by auto
ultimately have KGdisj: G ! i <#> K ∩ G ! (i + 1) = G ! i ∨ G ! i <#> K ∩ G ! (i + 1) = G ! (i + 1)
using Gimax unfolding max-normal-subgroup-def max-normal-subgroup-axioms-def by auto
obtain ϕ where ϕ ∈ iso (G\{carrier := K ∩ G ! (i + 1)} Mod (G ! i ∩ (K ∩ G ! (i + 1))))
(G\{carrier := G ! i <#> G\{carrier := G ! (i + 1)} K ∩ G ! (i + 1))
Mod G ! i

using second-isomorphism-grp.normal-intersection-quotient-isom
unfolding second-isomorphism-grp-def second-isomorphism-grp-axioms-def
using GiSi subKGSiGSi GiSi normal-imp-subgroup by fastforce
hence ϕ ∈ iso (G\{carrier := K ∩ G ! (i + 1)} Mod (K ∩ G ! (i + 1) ∩ G ! i))
(G\{carrier := G ! i <#> G\{carrier := G ! (i + 1)} K ∩ G ! (i + 1))
Mod G ! i

by (metis inf-commute)
hence ϕ ∈ iso (G\{carrier := K ∩ G ! (i + 1)} Mod (K ∩ (G ! (i + 1) ∩ G ! i))
(G\{carrier := G ! i <#> G\{carrier := G ! (i + 1)} K ∩ G ! (i + 1))
Mod G ! i

by (metis Int-assoc)
hence ϕ ∈ iso (G\{carrier := K ∩ G ! (i + 1)} Mod (K ∩ G ! i))
(G\{carrier := G ! i <#> G\{carrier := G ! (i + 1)} K ∩ G ! (i + 1))
Mod G ! i)
by (metis GSi' Int-absorb2 Int-commute)

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

\((G[\text{carrier} := G! i <\#> K \cap G! (i + 1)]) \text{ Mod} G! i)\) unfolding set-mult-def by auto

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

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

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

assume \(G! i <\#> K \cap G! Su)i = G! i\)

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

\((G[\text{carrier} := G! i <\#> K \cap G! (i + 1)]) \text{ Mod} G! i)\) by auto

moreover obtain \(\psi\) where \(\psi \in \text{iso} \ (G[\text{carrier} := G! i])\) \(\text{ Mod} (carrier (G[\text{carrier} := G! i])))\) unfolding group.self-factor-iso groupGi by force

ultimately obtain \(\pi\) where \(\pi \in \text{iso} \ (G[\text{carrier} := K \cap G! (i + 1)]) \text{ Mod} (K \cap G! i))\) \((G[\text{carrier} := \{1\}])\) using iso-set-trans by fanforce

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

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

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

using group.order-one-triv-iff KGsiKGgroup by blast

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

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

ultimately \(\text{have} \ K \cap G! i = \text{carrier} (G[\text{carrier} := K \cap G! (i + 1)])\) by (metis normal_fact-group-trivial-iff)

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

with \(j\) have \(\text{False using remdups-adj-adjacent KGnotempty Suc-eq-plus}1\) by metis

thus simple-group \((G[\text{carrier} := \text{remdups-adj} (\text{map} (\cap G! (j))) \text{!} j) \text{ Mod} \text{remdups-adj} (\text{map} (\cap G! (j))) \text{!} j)\).

next

assume \(G! i <\#> K \cap G! Su)i = G! i\)

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

\((G[\text{carrier} := G! i <\#> K \cap G! (i + 1)]) \text{ Mod} G! i)\) by auto

then obtain \(\varphi'\) where \(\varphi' \in \text{iso} \ (G[\text{carrier} := G! i (i + 1)]) \text{ Mod} G! i)\)

using KGsiKGgroup group.iso-set-sym by auto

with Gsimple KGsiKGsgroup have simple-group \((G[\text{carrier} := K \cap G! (i + 1)]) \text{ Mod} (K \cap G! i))\) by (metis simple-group.iso-simple)

with \(i\) show simple-group \((G[\text{carrier} := \text{remdups-adj} (\text{map} (\cap G! (j))) \text{!} j)\) suc
lemma (in group) composition-series-extend:
  assumes composition-series (G{carrier := H}) F
  assumes simple-group (G Mod H) H ≤ G
  shows composition-series (F @ [carrier G])
unfolding composition-series-def composition-series-axioms-def
proof auto
  from assms(1) interpret compF: composition-series G{carrier := H} F .
  show normal-series G (F @ [carrier G]) using assms(3) F.is-normal-series
  by (metis normal-series-extend)
  fix i
  assume i: i < length F
  show simple-group (G{carrier := (F @ [carrier G]) ! i}) Mod (F @ [carrier G]) ! i
    proof (cases i = length F - 1)
      case True
      hence (F @ [carrier G]) ! Suc i = carrier G by (metis i diff-Suc-1 lessE nth-append-length)
      moreover have (F @ [carrier G]) ! i = F ! i by (metis butlast-snoc i nth-butlast)
      hence (F @ [carrier G]) ! i = H using True last-conv-nth F.is-normal-series.compF.last
      ultimately show ?thesis using assms(2) by auto
    next
      case False
      hence Suc i < length F using i by auto
      hence (F @ [carrier G]) ! Suc i = F ! Suc i using nth-append by metis
      moreover from i have (F @ [carrier G]) ! Suc i = F ! Suc i using nth-append by metis
      ultimately show ?thesis using (Suc i < length F) F.is-normal-series.compF.simplefact
    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 0
  hence i = j by auto
  thus G ! i ⊆ G ! j by auto
next
  case (Suc k i j)
  hence i': i + (Suc k) = j + 1 < length G by auto
  hence i': i + 1 ≤ j by auto
  have G ! i ⊆ G ! (i + 1) using i' normal normal-imp-subgroup subgroup subset
  by force
  moreover have j - (i + 1) = k j < length G using Suc assms by auto
hence \( G ! (i + 1) \subseteq G ! j \) using \( \text{Suc}(1) \) \( ij \) by \text{auto}
ultimately show \( G ! i \subseteq G ! j \) by \text{simp}
qed

end

theory GroupIsoClasses
imports
  HOL-Algebra.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 \text{auto}
qed

definition group-iso-rel :: 'a group \Rightarrow 'a group \Rightarrow bool
where group-iso-rel G H = (\exists \phi. \phi \in \text{iso} (\text{Rep-group G}) (\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 iso-set-refl by blast
  qed
next
  show symp group-iso-rel
  proof (rule sympl)
    fix G H :: 'b group
    assume group-iso-rel G H
    then obtain \( \phi \) where \( \phi \in \text{iso} (\text{Rep-group G}) (\text{Rep-group H}) \)
    unfolding group-iso-rel-def by \text{auto}
    then obtain \( \phi' \) where \( \phi' \in \text{iso} (\text{Rep-group H}) (\text{Rep-group G}) \)
    using group.iso-sym
    Rep-group
    unfolding group.iso-set-sym by blast
    thus group-iso-rel H G unfolding group-iso-rel-def by \text{auto}
  qed
next
  show transp group-iso-rel
  proof (rule transpI)

46
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 ∈ iso 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 ∈ iso G H) = (∃ϕ. ϕ ∈ iso (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 ?thesis.
  qed

end

theory JordanHolder
imports
  CompositionSeries
  MaximalNormalSubgroups
  HOL-Library.Multiset
  GroupIsoClasses
begin
10 The Jordan-Hölder Theorem

locale jordan-hoelder = group
+ compHG?: composition-series \( G \triangleleft H \)
+ compSG?: composition-series \( G \triangleleft S \) for \( H \) and \( S \)
+ 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 \( G > 1 \)
  shows butlast quotients = normal-series.quotients (\( G[G\langle \text{carrier} := G \rangle(\text{length } G - 1) - 1) \)) \( (\text{take } (\text{length } G - 1) G) \)
  proof (rule nth-equalityI)
    define n where \( n = \text{length } G - 1 \)
    hence \( n = \text{length } (\text{take } n G) \) \( n > 0 \)
    have \( n < \text{length } G \) using assms notempty by auto
    interpret normal\( G \) butlast: normal-series \( G[G\langle \text{carrier} := G \rangle(\text{length } G - n - 1)] \) \( \text{take } n G \)

    using normal-series-prefix-closed \( \langle n > 0 \rangle \langle n < \text{length } G \rangle \) by auto
    have length \( (\text{butlast quotients}) = \text{length quotients} - 1 \) by (metis length-butlast)
    also have \( \ldots = \text{length } G - 1 - 1 \) by (metis add-diff-cancel-right' quotients-length)
    also have \( \ldots = \text{length } (\text{take } n G) - 1 \) by (metis \( (n = \text{length } (\text{take } n G) \) n-def)
    also have \( \ldots = \text{length normal}\( G \) butlast.quotients by (metis normal\( G \) butlast.quotients-length \( \text{diff-add-inverse}\))
    finally show length \( (\text{butlast quotients}) = \text{length normal}\( G \) butlast.quotients \).
    have \( \forall i < \text{length } (\text{butlast quotients}) \). butlast quotients \( i = \text{normal}\( G \) butlast.quotients \) ! i
    proof auto
      fix i
      assume \( i : i < \text{length quotients} = \text{Suc } 0 \)
      hence \( i':i < \text{length } G - 1 \)
      unfolding \( \text{length quotients} - \text{n unfolding n-def} \) using quotients-length by auto
      from i have butlast quotients \( i = \text{quotients} \) ! i by (metis \( \text{One-nat-def} \)
 len-butlast nth-butlast)
      also have \( \ldots = G[G\langle \text{carrier} := G \rangle(\text{length } G - n - 1)] \) \( \text{Mod } G \) ! i unfolding quotients-def
      using \( i'(1) \) by auto
      also have \( \ldots = G[G\langle \text{carrier} := (\text{take } n G) \rangle(\text{length } G - n - 1)] \) \( \text{Mod } (\text{take } n G) \) ! i using
      \( i'(2,3) \) nth-take by metis
      also have \( \ldots = \text{normal}\( G \) butlast.quotients \) ! i unfolding normal\( G \) butlast.quotients-def
      using \( i' \) by fastforce
      finally show butlast \( (\text{normal-series.quotients } G G) \) ! i = normal-series.quotients
      \( (G[G\langle \text{carrier} := G \rangle(\text{length } G - n - 1)] - 1)] \) \( \text{take } n G \) ! i by auto
      qed
      thus \( \forall i. i < \text{length } (\text{butlast quotients}) \)
      unfolding quotients = normal-series.quotients \( G[G\langle \text{carrier} := G \rangle(\text{length } G - n - 1)] - 1)] \) \( \text{take } (\text{length } G - 1) G \) ! i
      qed
The main part of the Jordan-Holder 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 $G \mathcal{S}$
- Assumes composition-series $G \mathcal{H}$

**Shows** $\text{mset (map group.iso-class (normal-series.quotients G \mathcal{S}))} = \text{mset (map group.iso-class (normal-series.quotients G \mathcal{H}))}$

**Using** assms

**Proof** (induction length $\mathcal{S}$ arbitrary; $\mathcal{S} \neq \mathcal{H}$ rule: full-nat-induct)

- Case $(\mathcal{S} \neq \mathcal{H}) (G)$
  - Then interpret $\text{comp}\mathcal{S}$: composition-series $G \mathcal{S}$ by simp
  - From 1 interpret $\text{comp}\mathcal{H}$: composition-series $G \mathcal{H}$ by simp

- Show $\ ? \text{case}$
  - Proof (cases length $\mathcal{S} \leq 2$)
    - Next
      - Case True
        - Hence length $\mathcal{S} = 0 \lor \text{length } \mathcal{S} = 1 \lor \text{length } \mathcal{S} = 2$ by arith
      - With $\text{comp}\mathcal{S}.\text{notempty}$ have length $\mathcal{S} = 1 \lor \text{length } \mathcal{S} = 2$ by simp
    - Thus $\ ? \text{thesis}$
      - Proof (auto simp del: mset-map)
        - First trivial case: $\mathcal{S}$ is the trivial group.
          - Assume length $\mathcal{S} = \text{Suc 0}$
          - Hence length:length $\mathcal{S} = 1$ by simp
          - Hence length [] + 1 = length $\mathcal{S}$ by auto
        - Moreover from length have $\text{char}\mathcal{S} = \{1\_G\}$ by (metis $\text{comp}\mathcal{S}.\text{composition-series-length-one}$)
          - Hence carrier $G = \{1\_G\}$ by (metis $\text{comp}\mathcal{S}.\text{composition-series-triv-group}$)
        - With length $\text{char}\mathcal{S}$ have $\mathcal{S} = \mathcal{H}$ using $\text{comp}\mathcal{H}.\text{composition-series-triv-group}$
      - By simp
        - Thus $\ ? \text{thesis by simp}$
    - Next
      - Second trivial case: $\mathcal{S}$ is simple.
        - Assume length $\mathcal{S} = 2$
        - Hence $\text{char}\mathcal{S} = \{1\_G\}$, carrier $G$ by (metis $\text{comp}\mathcal{S}.\text{length-two-unique}$)
        - Hence simple:composition-series $G$ by (metis $\text{comp}\mathcal{S}.\text{composition-series-simple-group}$)
        - Hence $\mathcal{H} = \{1\_G\}$, carrier $G$ using $\text{comp}\mathcal{H}.\text{composition-series-simple-group}$
      - By auto
        - With $\text{char}$ have $\mathcal{S} = \mathcal{H}$ by simp
        - Thus $\ ? \text{thesis by simp}$
    - Qed
  - Next
    - Case False
— Non-trivial case: $\mathfrak{G}$ has length at least 3.

**hence** \( \text{length:length } \mathfrak{G} \geq 3 \) by simp

— First we show that $\mathfrak{H}$ must have a length of at least 3.

**hence** \( G \) using \( \text{comp}\mathfrak{G}.\text{composition-series-simple-group} \) by auto

**hence** $\mathfrak{H} \neq \{1_G\}$, carrier $G$ using \( \text{comp}\mathfrak{H}.\text{composition-series-simple-group} \) by auto

**hence** length $\mathfrak{H} \neq 2$ using \( \text{comp}\mathfrak{H}.\text{length-two-unique} \) by auto

moreover from length have carrier $G \neq \{1_G\}$ using \( \text{comp}\mathfrak{G}.\text{composition-series-length-one} \) comp\mathfrak{G}.\text{composition-series-triv-group} by auto

**hence** length $\mathfrak{H} \neq 1$ using \( \text{comp}\mathfrak{H}.\text{composition-series-length-one} \) comp\mathfrak{H}.\text{composition-series-triv-group} by auto

moreover from \( \text{comp}\mathfrak{H}.\text{notempty} \) have length $\mathfrak{H} \neq 0$ by simp

ultimately have length$\exists m: m \geq 3$ using \( \text{comp}\mathfrak{H}.\text{notempty} \) by arith

define $m$ where $m = \text{length } \mathfrak{H} - 1$

define $n$ where $n = \text{length } \mathfrak{G} - 1$

from length$\exists m: m > 0$ m < length $\mathfrak{H}$ \((m - 1) + 1 < \text{length } \mathfrak{H} m - 1 = \text{length } \mathfrak{H} - 2 m - 1 + 1 = \text{length } \mathfrak{H} - 1 m - 1 < \text{length } \mathfrak{H}$

unfolding $m$-def by auto

from length have $n': n > 0$ n < length $\mathfrak{G} (n - 1) + 1 < \text{length } \mathfrak{G} n - 1 < \text{length } \mathfrak{G}$ Suc $n \leq \text{length } \mathfrak{G}$

$\text{n - 1 = length } \mathfrak{G} - 2 n - 1 + 1 = \text{length } \mathfrak{G} - 1$ unfolding $n$-def by auto

define $\mathfrak{G} \text{\textit{Pn}}$ where $\mathfrak{G} \text{\textit{Pn}} = G[\text{carrier} := \mathfrak{G} ! (n - 1)]$

define $\mathfrak{H} \text{\textit{Pm}}$ where $\mathfrak{H} \text{\textit{Pm}} = G[\text{carrier} := \mathfrak{H} ! (m - 1)]$

then interpret grp$\mathfrak{G} \text{\textit{Pn}}: \text{group } \mathfrak{G} \text{\textit{Pn}}$ unfolding $\mathfrak{G} \text{\textit{Pn}}$-def using $n'$ by (metis comp\mathfrak{G}.\text{normal-series-subgroups} comp\mathfrak{G}.\text{subgroup-imp-group})

interpret grp$\mathfrak{H} \text{\textit{Pm}}: \text{group } \mathfrak{H} \text{\textit{Pm}}$ unfolding $\mathfrak{H} \text{\textit{Pm}}$-def using $m'$ comp\mathfrak{H}.\text{normal-series-subgroups} 1(2) group."group-imp-group" by force

have fin$\text{\textit{Gbl}}:\text{finite } (\text{carrier } \mathfrak{G} \text{\textit{Pn}})$ using \((n - 1 < \text{length } \mathfrak{G} 1(3)\) unfolding $\mathfrak{G} \text{\textit{Pn}}$-def using comp\mathfrak{G}.\text{normal-series-subgroups} comp\mathfrak{G}.\text{subgroup-finite} by auto

have fin$\text{\textit{Hbl}}:\text{finite } (\text{carrier } \mathfrak{H} \text{\textit{Pm}})$ using \((m - 1 < \text{length } \mathfrak{H} 1(3)\) unfolding $\mathfrak{H} \text{\textit{Pm}}$-def using comp\mathfrak{G}.\text{normal-series-subgroups} comp\mathfrak{G}.\text{subgroup-finite} by auto

have quot$\mathfrak{G}$\text{\textit{notempty}}:\text{comp}\mathfrak{G}.\text{quotients} \# \[ \] using comp\mathfrak{G}.\text{quotients-length-length} by auto

have quot$\mathfrak{H}$\text{\textit{notempty}}:\text{comp}\mathfrak{H}.\text{quotients} \# \[ \] using comp\mathfrak{H}.\text{quotients-length-length}$\exists m: m > 0$ by auto

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

interpret $\mathfrak{H} \text{\textit{butlast}}$: \text{composition-series } \mathfrak{H} \text{\textit{Pm}} take $m$ $\mathfrak{H}$ using comp\mathfrak{H}.\text{composition-series-prefix-closed} m'(1,2) $\mathfrak{H} \text{\textit{Pm}}$-def by auto

interpret $\mathfrak{G} \text{\textit{butlast}}$: \text{composition-series } $\mathfrak{G} \text{\textit{Pn}}$ take $n$ $\mathfrak{G}$ using comp\mathfrak{G}.\text{composition-series-prefix-closed} n'(1,2) $\mathfrak{G} \text{\textit{Pn}}$-def by auto

have itaken$\exists n = \text{length } (\text{take } n \mathfrak{G})$ using length-take n'(2) by auto

have itaken$\exists m = \text{length } (\text{take } m \mathfrak{H})$ using length-take m'(2) by auto

show $?\text{thesis}$

proof (cases $\mathfrak{H} ! (m - 1) = \mathfrak{G} ! (n - 1)$)

— If $\mathfrak{H} ! (l - 1) = \mathfrak{G} ! 1$, everything is simple...

case True

— The last quotients of $\mathfrak{G}$ and $\mathfrak{H}$ are equal.

have lasteq:last comp\mathfrak{G}.quotients = last comp\mathfrak{H}.quotients
proof

  from length have by:length \G - 1 - 1 + 1 = length \G - 1 by (metis Suc-diff-1 Suc-eq-plus1 n'(1) n-def)
  from lengthSucbig have by:length \G - 1 - 1 + 1 = length \G - 1 by (metis Suc-diff-1 Suc-eq-plus1 (0 < m) m-def)
  have last compSuc.quotients = \G Mod \G ! (n - 1) using length compSuc.last-quotient

unfolding n-def by auto
  also have \ldots = \G Mod \G ! (m - 1) using True by simp
  also have \ldots = last compSuc.quotients using lengthSucbig compSuc.last-quotient

unfolding m-def by auto
  finally show ?thesis .

qed

from \texttt{laken} have ind:mset (map group.iso-class \S bullast.quotients) = mset (map group.iso-class \S bullast.quotients)
  using \texttt{I (I) True n'(5) grp\S Pn.is-group finGbl \S bullast.is-composition-series \S bullast.is-composition-series unfolding \S Pn-def \S Pm-def by metis
  have mset (map group.iso-class \S bullast.quotients) = mset (map group.iso-class (butlast compSuc.quotients \ S [last (compSuc.quotients)])) by (simp add: quots\S notempty)
  also have \ldots = mset (map group.iso-class \S bullast.quotients \ S [last (compSuc.quotients)])) using compSuc.quotients-bullast length unfolding n-def \& Pn-def

by auto
  also have \ldots = mset ((map group.iso-class \S bullast.quotients) \ S [group.iso-class
  (last (compSuc.quotients))]) by auto
  also have \ldots = mset (map group.iso-class \S bullast.quotients) + \{ \# group.iso-class
  (last (compSuc.quotients)) \} by auto
  also have \ldots = mset (map group.iso-class \S bullast.quotients) + \{ \# group.iso-class
  (last (compSuc.quotients)) \} using ind by simp
  also have \ldots = mset (map group.iso-class \S bullast.quotients) + \{ \# group.iso-class
  (last (compSuc.quotients)) \} using lasteq by simp
  also have \ldots = mset ((map group.iso-class \S bullast.quotients) \ S [group.iso-class
  (last (compSuc.quotients))]) by auto
  also have \ldots = mset (map group.iso-class \S bullast.quotients \ S [last (compSuc.quotients)]) by auto
  also have \ldots = mset (map group.iso-class \S bullast.quotients \ S [last (compSuc.quotients)]) using lengthSucbig compSuc.quotients-bullast unfolding m-def

\S Pm-def by auto
  also have \ldots = mset (map group.iso-class \S bullast.quotients) using append-bullast-last-id quots\S notempty by simp

finally show ?thesis .

next

case False

  define \S PmInt\S Pn where \S PmInt\S Pn = \G[carrier := \S ! (m - 1) \cap \G ! (n - 1)]

  interpret \G Pnmax: max-normal-subgroup \G ! (n - 1) \G unfolding n-def
  by (metis add-lessD1 diff-diff-add \n'(3) add.commute one-add-one I(3) \comp \S, snd-to-last-max-normal)

  interpret \S Pmmax: max-normal-subgroup \S ! (m - 1) \S unfolding m-def
  by (metis add-lessD1 diff-diff-add \m'(3) add.commute one-add-one I(3)
have $\mathcal{H} PnmaxG: \mathcal{H} ! (m - 1) \triangleleft G$ using $\mathcal{H} PnmaxG.normal-series-snd-to-last m'(4)$ unfolding m-def by auto

have $\mathcal{H} PnmaxG: \mathcal{H} ! (n - 1) \triangleleft G$ using $\mathcal{H} PnmaxG.normal-series-snd-to-last n'(6)$ unfolding n-def by auto

have $\mathcal{H} Pm\mathcal{H} : PnmaxG: \mathcal{H} ! (m - 1) \cap \mathcal{H} ! (n - 1) \triangleleft G$ using $\mathcal{H} PnmaxG$

$\mathcal{H} PnmaxG$ by (rule $\mathcal{H} PnmaxG.normal-subgroup-intersect)

have Intnorm$\mathcal{H} Pn: \mathcal{H} ! (m - 1) \cap \mathcal{H} ! (n - 1) \triangleleft \mathcal{H} Pn$ using $\mathcal{H} PnmaxG$

$\mathcal{H} PnmaxG$ Int-lower2 unfolding $\mathcal{H} Pn$-def

by (metis $\mathcal{H} PnmaxG.normal-subgroup-intersect compG.normal-subgroup-intersect n'(4))$

then interpret grp$\mathcal{H} PnMod\mathcal{H} Pm\mathcal{H}: $group $\mathcal{H} Pn Mod \mathcal{H} ! (m - 1) \cap \mathcal{H} ! (n - 1)$ by (rule normal.factorgroup-is-group)

have Intnorm$\mathcal{H} Pn: \mathcal{H} ! (m - 1) \cap \mathcal{H} ! (n - 1) \triangleleft \mathcal{H} Pm$ using $\mathcal{H} PnmaxG$

$\mathcal{H} PnmaxG$ Int-lower2 Int-commute unfolding $\mathcal{H} Pn$-def

by (metis $\mathcal{H} PnmaxG.normal-subgroup-intersect compG.normal-series-subgroups m'(6))$

then interpret grp$\mathcal{H} Pm\mathcal{H}: $group $\mathcal{H} Pm Mod \mathcal{H} ! (m - 1) \cap \mathcal{H} ! (n - 1)$ by (rule normal.factorgroup-is-group)

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

have $\mathcal{H} PmSub\mathcal{H} Pn:- (\mathcal{H} ! (m - 1) \subseteq \mathcal{H} ! (n - 1))$ using $\mathcal{H} PnmaxG.max-normal$ $\mathcal{H} PnmaxG False[$symmetric$]$ $\mathcal{H} PnmaxG.proper$ by simp

have $\mathcal{H} PmSub\mathcal{H} Pn:- (\mathcal{H} ! (n - 1) \subseteq \mathcal{H} ! (m - 1))$ using $\mathcal{H} PnmaxG.max-normal$ $\mathcal{H} PnmaxG False$ $\mathcal{H} PnmaxG.proper$ by simp

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

have $\mathcal{H} PmSub\mathcal{H} Pnsetmult: \mathcal{H} ! (m - 1) \subseteq \mathcal{H} ! (n - 1)$ $\mathcal{H} ! (n - 1)$

using second-isomorphism-grp.$H$-contained-in-set-mult $\mathcal{H} PnmaxG.is-normal$ $\mathcal{H} PnmaxG.normal-imp-subgroup$

unfolding second-isomorphism-grp-def second-isomorphism-grp-axioms-def max-normal-subgroup-def by metis

have $\mathcal{H} PnSub\mathcal{H} Pnsetmult: \mathcal{H} ! (n - 1) \subseteq \mathcal{H} ! (m - 1)$ $\mathcal{H} ! (n - 1)$

using second-isomorphism-grp.$S$-contained-in-set-mult $\mathcal{H} PnmaxG.is-normal$ $\mathcal{H} PnmaxG.normal-imp-subgroup$

unfolding second-isomorphism-grp-def second-isomorphism-grp-axioms-def max-normal-subgroup-def by metis

have $\mathcal{H} ! (n - 1) \neq (\mathcal{H} ! (m - 1))$ $\mathcal{H} ! (n - 1)$ using $\mathcal{H} PmSub\mathcal{H} Pnsetmult$

not$\mathcal{H} PmSub\mathcal{H} Pn$ by auto

hence set-mult$G:(\mathcal{H} ! (m - 1))$ $\mathcal{H} ! (n - 1)) = carrier G$

using $\mathcal{H} PnmaxG.max-normal$ $\mathcal{H} PnmaxG.is-normal$ $\mathcal{H} PnmaxG.compG.normal-subgroup-set-mult-closed$

$\mathcal{H} PnSub\mathcal{H} Pnsetmult$ by metis

then obtain $\phi$ where $\phi \in iso (\mathcal{H} Pn Mod (\mathcal{H} ! (m - 1) \cap \mathcal{H} ! (n - 1)))$

$\mathcal{H} PnSub\mathcal{H} Pn$ by (carrier $G)$ $\mathcal{H} Pn Mod (\mathcal{H} ! (m - 1))$

using second-isomorphism-grp.$normal-intersection-quotient-isom$ $\mathcal{H} PnmaxG$

$\mathcal{H} PnmaxG.is-normal$ normal-imp-subgroup

unfolding second-isomorphism-grp-def second-isomorphism-grp-axioms-def max-normal-subgroup-def $\mathcal{H} Pn$-def by metis

hence $\phi: \phi \in iso (\mathcal{H} Pn Mod (\mathcal{H} ! (m - 1) \cap \mathcal{H} ! (n - 1))) (G Mod \mathcal{H} ! (m - 1) \cap \mathcal{H} ! (n - 1)))$ (G Mod \mathcal{H} ! (m - 1) \cap \mathcal{H} ! (n - 1)))
\( \text{\textit{by auto}} \)

\( \text{then obtain } \varphi_2 \text{ where } \varphi_2 : \varphi_2 \in \text{iso } (G \text{ Mod } \mathfrak{H} \setminus (m - 1)) (\mathfrak{G} \text{ Pm Mod } (\mathfrak{H} \setminus (m - 1))) \text{ by auto} \)

\( \text{using group.iso-set-sgm grp}\mathfrak{G} \text{ Pm Mod}\mathfrak{H} \text{ PmInt}\mathfrak{G} \text{ is-group by auto} \)

\( \text{moreover have simple-group } (G\{\text{carrier }:= \mathfrak{H} \setminus (m - 1 + 1)} ) \text{ Mod } \mathfrak{H} \setminus (m - 1)) \text{ by using comp}\mathfrak{H} \text{. last last-cons-nth comp}\mathfrak{H} \text{. notempty m'(5) by fastforce} \)

\( \text{ultimately have simple}\mathfrak{G} \text{ Pm ModInt : simple-group } (\mathfrak{G} \text{ Pm Mod } (\mathfrak{H} \setminus (m - 1)) \cap \mathfrak{G} \setminus (n - 1))) \text{ by (metis } \mathfrak{H} \text{ Pmnorm}\mathfrak{G} \text{ normal factorgroup-is-group} \)

---

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

\( \text{have } \mathfrak{H} \text{ PmSubSetmult'} \mathfrak{H} \setminus (m - 1) \subseteq \mathfrak{G} \setminus (n - 1) \text{ by simp} \)

\( \text{using second-isomorphism-grp.S-contained-in-set-mult } \mathfrak{G} \text{ Pmmax.is-normal } \mathfrak{H} \text{ Pmnorm}\mathfrak{G} \text{ normal-imp-subgroup} \)

\( \text{unfolding second-isomorphism-grp-def second-isomorphism-grp-axioms-def } \text{max-normal-subgroup-def by metis} \)

\( \text{have } \mathfrak{G} \text{ PmSubSetmult'} \mathfrak{G} \setminus (n - 1) \subseteq \mathfrak{G} \setminus (n - 1) \text{ by simp} \)

\( \text{using second-isomorphism-grp.H-contained-in-set-mult } \mathfrak{G} \text{ Pmmax.is-normal } \mathfrak{H} \text{ Pmnorm}\mathfrak{G} \text{ normal-imp-subgroup} \)

\( \text{unfolding second-isomorphism-grp-def second-isomorphism-grp-axioms-def } \text{max-normal-subgroup-def by metis} \)

\( \text{have } \mathfrak{H} \setminus (m - 1) \neq (\mathfrak{G} \setminus (n - 1)) \text{ by simp} \)

\( \text{using } \mathfrak{G} \text{ PmSubSetmult'} \text{ by metis} \)

\( \text{from set-mult } G \text{ obtain } \psi \text{ where } \psi \in \text{iso } (\mathfrak{G} \text{ Pm Mod } (\mathfrak{H} \setminus (n - 1) \cap \mathfrak{H} \setminus (m - 1))) \text{ by simp} \)

\( \text{using second-isomorphism-grp.normal-intersection-quotient-isom } \mathfrak{G} \text{ Pmmax.is-normal } \text{normal-imp-subgroup} \)

\( \text{unfolding second-isomorphism-grp-def second-isomorphism-grp-axioms-def } \text{max-normal-subgroup-def by metis} \)

\( \text{have } \psi : \psi \in \text{iso } (\mathfrak{G} \text{ Pm Mod } (\mathfrak{H} \setminus (n - 1) \cap (\mathfrak{G} \setminus (n - 1)))) \text{ by simp} \)

\( \text{ultimately have simple}\mathfrak{G} \text{ Pm ModInt : simple-group } (\mathfrak{G} \text{ Pm Mod } (\mathfrak{H} \setminus (m - 1)) \cap \mathfrak{G} \setminus (n - 1))) \text{ by (metis } \mathfrak{H} \text{ Pmnorm}\mathfrak{G} \text{ normal factorgroup-is-group} \)
\[ \cap \mathcal{G} ! (n - 1) \]

using simple-group.iso-simple \( \text{grp}\{ \text{Pm}\{ \text{Mod}\{ \text{G}\{ \text{Pn}\{ \text{is-group} \}} \} \} \} \) by auto

interpret \( \text{grp}\{ \text{G}\{ \text{Mod}\{ \text{G}\{ \text{Pn}\{ \text{is-group} \}} \} \} \} \) using \( \text{G}\{ \text{Mod}\{ \text{G}\{ \text{Pn}\{ \text{is-group} \}} \} \} \) normal, factorgroup-is-group

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

define \( \mathcal{R} \) where \( \mathcal{R} = \text{remdups-adj} (\text{map} \ (\cap \ (\mathcal{G} ! (n - 1))) \ \mathcal{G}) \)
define \( \mathcal{L} \) where \( \mathcal{L} = \text{remdups-adj} (\text{map} \ (\cap \ (\mathcal{G} ! (n - 1))) \ \mathcal{G}) \)

interpret \( \mathcal{R} \): composition-series \( \text{Pm}\{ \text{G}\{ \text{mod}\{ \text{G}\{ \text{Pn}\{ \text{is-group} \}} \} \} \} \) unfolding \( \mathcal{R}\)-def by auto

interpret \( \mathcal{L} \): composition-series \( \text{G}\{ \text{Pn}\{ \text{is-group} \}} \) unfolding \( \mathcal{L}\)-def by auto

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

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

hence multisets\( \text{Gblast} \mathcal{L}: \text{mset} \) (map group.iso-class \( \text{Gblast}\).quotients) =

\[ \text{mset} \ (\text{map} \ (\text{group}.iso-class \ \text{Gblast}.\text{quotients}) \) using \(1\).hyps \( \text{grp}\{ \text{G}\{ \text{Pn}\{ \text{is-group} \}} \} \text{finGb} \) \( \text{Gblast}\).is-composition-series \( \mathcal{L} \).is-composition-series by metis

hence \( \text{length}\mathcal{L}: n = \text{length} \ \mathcal{L} \) using \( \text{Gblast}.\text{quotients-}\text{length} \ \mathcal{L} \).quotients-\text{length} \text{length-map size-mset taken} \) by metis

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

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

have \( \text{Inteq}\{ \text{sndlast}\}: \mathcal{G} ! (m - 1) \cap \mathcal{G} ! (n - 1) = \mathcal{L} ! (\text{length} \ (\mathcal{L} - 1 - 1)) \)

proof –

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

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

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

\[ \mathcal{L} ! (\text{length} (\mathcal{L} - 1 - 1) = (\text{map} \ (\cap) (\mathcal{G} ! (n - 1))) \) \( \mathcal{H} ! (i + 1) \)

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

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

by (metis \(\text{Suc-diff-1}\) \(\text{Suc-eq-plus1}\)

hence \( \mathcal{G}\{ \text{Pm}\{ \text{sub}\{ \text{Pm}\{ \text{G}\{ \text{Pn}\{ \text{is-group} \}} \} \} \} \} ! (n - 1) \subseteq \mathcal{H} ! (i + 1) \)

using \( \text{Glast} . \text{notempty} \) unfolding \( \mathcal{G}\{ \text{Pm}\{ \text{def}\} \} \) by auto

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

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

not\( \mathcal{G}\{ \text{Pn}\{ \text{Sub}\{ \text{Pm}\{ \text{Pn}\{ \text{Sub}\} \} \} \} \} \) by fastforce

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

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

qed

hence \( \text{sndlast}\): \( \mathcal{G}\{ \text{Pm}\{ \text{Int}\{ \text{G}\{ \text{Pn}\{ \text{is-group} \}} \} \} \} = (\mathcal{G}\{ \text{Pn}\{ \text{carrier} := \mathcal{L} ! (\text{length} \ (\mathcal{L} - 1 - 1)) \}) \)

unfolding \( \mathcal{G}\{ \text{Pm}\{ \text{def}\} \} \) by auto

then interpret \( \text{Gblast} : \text{composition-series} \mathcal{H}\{ \text{Pm}\{ \text{Int}\{ \text{G}\{ \text{Pn}\{ \text{is-group} \}} \} \} \} \) take \( \text{length} \ \mathcal{L} -
1) \( \mathcal{L} \) using \( \text{length}\mathcal{L}' \) \( \text{composition-series-prefix-closed by \text{metis}} \) from \( \text{length } \mathcal{L} > 1 \) have \( \text{quot}\mathcal{L}'\text{notemtpy}\mathcal{L} \).\text{quotients} \( \neq \) \( \emptyset \) unfolding \( \mathcal{L}'\text{quotients-def by \text{auto}} \)

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

have \( \text{length } \mathcal{R} > 1 \)

proof (rule \text{ccontr})

assume \( \neg \text{length } \mathcal{R} > 1 \)

with \( \mathcal{R} \text{.notempty} \) have \( \text{length } \mathcal{R} = 1 \) by (metis \text{One-nat-def Suc-lessI length-greater-0-conv})

hence carrier \( \mathcal{S}_{Pm} = \{1_{\mathcal{S}_{Pm}}\} \) using \( \mathcal{R}'\text{composition-series-length-one} \)

\( \mathcal{R}'\text{composition-series-triv-group by \text{auto}} \)

hence carrier \( \mathcal{S}_{Pm} = \{1_{\mathcal{G}}\} \) unfolding \( \mathcal{S}_{Pm\text{-def by \text{auto}} \} \)

hence carrier \( \mathcal{S}_{Pm} \subseteq \mathcal{G} \) \( (n - 1) \) using \( \mathcal{G}_{\text{Pmax.is-subgroup}} \)

\( \mathcal{S}_{Pm}\text{-one-closed by \text{auto}} \)

with \( \mathcal{S}_{\text{notempty PmSub}\mathcal{S}_{Pn}} \) show False unfolding \( \mathcal{S}_{Pm\text{-def by \text{auto}} \} \)

qed

hence \( \text{length}\mathcal{R}'\text{.length } \mathcal{R} - 1 > 0 \) \( \text{length } \mathcal{R} - 1 \leq \text{length } \mathcal{R} \) by \text{auto}

have \( \text{Inteq}\mathcal{R}_{\text{sndlast}}\mathcal{H} \) \((m - 1) \cap \mathcal{G} \) \( (n - 1) = \mathcal{R} \) \((\text{length } \mathcal{R} - 1 - 1) \)

proof —

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

moreover have \( \mathcal{K}_{\text{Gnotempty}}\)\((\mathcal{G}) \) \( (m - 1) \) \( (n - 1) \) \( \mathcal{G} \) \( (\text{length } \mathcal{R} - 1 - 1) \) \( \mathcal{R} \)

using \( \text{comp}\mathcal{G}_{\text{notempty by}} \) \( \text{(metis \text{Nil-is-map-cone})} \)

ultimately obtain \( i \) where \( i: i + 1 < \text{length} \) \( (\mathcal{G}) \) \( (m - 1) \) \( \mathcal{G} \)

\( \mathcal{R} ! (\text{length } \mathcal{R} - 1 - 1) = (\mathcal{G}) \) \( (m - 1) \) \( \mathcal{G} \) \( (\text{length } \mathcal{R} - 1 - 1 + 1) \)

using \( \text{rmdups-adj-obtain-adjacency unfolding } \mathcal{R}_{\text{def by force}} \)

hence \( \mathcal{R} ! (\text{length } \mathcal{R} - 1 - 1) = \mathcal{G} ! (i + 1) \cap \mathcal{H} ! (m - 1) \) \( \mathcal{R} ! (\text{length } \mathcal{R} - 1 - 1 + 1) \)

by \( \text{(metis } \text{Suc-diff-1 Suc-lessI}) \)

hence \( \mathcal{S}_{\text{Pmsub}\mathcal{G}_{Pn}}\mathcal{S}_{Pm} ! (m - 1) \) \( \mathcal{G} ! (i + 1) \) \( (m - 1) \) \( \mathcal{R}_{\text{last } \mathcal{R}_{\text{notempty \text{last-conv-nth unfolding } \mathcal{S}_{Pm\text{-def by \text{auto}} \} \) \) \( \mathcal{S}_{\text{notempty PmSub}\mathcal{G}_{Pn}} \) \( \mathcal{S}_{\text{Pmsub}\mathcal{G}_{Pn}} \) \( \mathcal{S}_{\text{notempty PmSub}\mathcal{G}_{Pn}} \) \( \text{by fastforce} \)

ultimately have \( n - 1 = i \) by \text{auto}

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

qed

have composition-series \((G\{\text{carrier} := \mathcal{R} ! (\text{length } \mathcal{R} - 1 - 1)\})\) \( \text{(take } (\text{length } \mathcal{R} - 1 - 1) \) \( \mathcal{R} \)

using \( \text{length}\mathcal{R}' \) \( \mathcal{R}'\text{.composition-series-prefix-closed unfolding } \mathcal{S}_{\text{PmInt}\mathcal{G}_{Pn}}\mathcal{S}_{Pm\text{-def}} \)

\( \mathcal{S}_{\text{Pm-def by fastforce}} \)

then interpret \( \mathcal{R}_{\text{butlast}: \text{composition-series } \mathcal{S}_{\text{PmInt}\mathcal{G}_{Pn}} \) \( \text{(take } (\text{length } \mathcal{R} - 1) \) \( \mathcal{R} \)

using \( \text{Inteq}\mathcal{R}_{\text{sndlast unfolding } \mathcal{S}_{\text{PmInt}\mathcal{G}_{Pn}}\mathcal{S}_{Pm\text{-def by \text{auto}} \} \)

from \( \text{FinGbl} \) have \( \text{finInt\text{-finite } (\text{carrier } \mathcal{S}_{\text{PmInt}\mathcal{G}_{Pn}})\) unfolding \( \mathcal{S}_{\text{PmInt}\mathcal{G}_{Pn}}\mathcal{S}_{Pm\text{-def by \text{simp}} \)

moreover have \( \text{Suc } (\text{take } (\text{length } \mathcal{L} - 1) \) \( \mathcal{L} \)) \( \leq \text{length } \mathcal{G} \) using \( \text{length}\mathcal{L} \) unfolding \( \text{n-def using } n'\text{.2 by \text{auto}} \)

55
ultimately have multisets \( \Sigma \) butlast : mset (map group.iso-class butlast.quotients) = mset (map group.iso-class \( \varnothing \) butlast.quotients)

using 1.hyps \( \Sigma \) butlast.is-group butlast.is-composition-series butlast.is-composition-series by auto

hence length (take (length \( \mathcal{K} \) − 1) \( \mathcal{K} \)) = length (take (length \( \mathcal{L} \) − 1) \( \mathcal{L} \))

using butlast.quotients-length \( \Sigma \) butlast.quotients-length length-map size-mset by metis

hence length (take (length \( \mathcal{K} \) − 1) \( \mathcal{K} \)) = \( n - 1 \) using length \( \mathcal{L} \) \( n'(1) \) by auto

hence length \( \mathcal{K} \).length \( \mathcal{K} \) = \( n \) by (metis Suc-diff-1 \( \mathcal{K} \).notempty butlast-conv-take length-butlast length-greater-0-conv \( n'(1) \))

— Apply the induction hypothesis on \( \mathcal{K} \) and \( \Sigma \) butlast

from Intergndlast have \( \mathcal{K} \) ndlast : \( \Sigma \) PnInt\( \mathcal{G} \) Pn = (\( \Sigma \) Pm[\( \text{carrier} := \mathcal{K} \) ! (length \( \mathcal{K} \) − 1 − 1)]) unfolding \( \Sigma \) PmInt\( \mathcal{G} \) Pn-def \( \Sigma \) Pm-def \( \mathcal{K} \).def by auto

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

hence multisets \( \Sigma \) butlast \( \mathcal{K} \).mset (map group.iso-class butlast.quotients) = mset (map group.iso-class \( \mathcal{K} \).quotients)

using 1.hyps \( \Sigma \) grp\( \mathcal{G} \) Pn.is-group finHbl butlast.is-composition-series \( \mathcal{K} \).is-composition-series by metis

hence length\( \mathcal{K} \).m = length \( \mathcal{K} \) using butlast.quotients-length \( \mathcal{K} \).quotients-length length-map size-mset butlast by metis

hence length \( \mathcal{K} \) > 1 length \( \mathcal{K} \) − 1 > 0 length \( \mathcal{K} \) − 1 \( \leq \) length \( \mathcal{K} \) using \( m'(4) \)

length\( \Sigma \) big by auto

hence quotes\( \Sigma \) notemtpy \( \mathcal{K} \).quotients \( \neq \) [] unfolding \( \mathcal{K} \).quotients-def by auto

interpret \( \mathcal{K} \) butlastadd\( \mathcal{G} \) Pn : composition-series \( \mathcal{G} \) Pn (take (length \( \mathcal{K} \) − 1) \( \mathcal{K} \)) @ [\( \mathcal{G} \) ! (\( n - 1 \))] using grp\( \mathcal{G} \) Pn.composition-series-extend butlast.is-composition-series simple\( \mathcal{G} \) PnModInt Intnorm\( \mathcal{G} \) Pn unfolding \( \mathcal{G} \) Pn-def \( \Sigma \) PmInt\( \mathcal{G} \) Pn-def by auto

interpret \( \Sigma \) butlastadd\( \mathcal{G} \) Pn : composition-series \( \mathcal{G} \) Pn (take (length \( \mathcal{L} \) − 1) \( \mathcal{L} \)) @ [\( \mathcal{G} \) ! (\( m - 1 \))] using grp\( \mathcal{G} \) Pn.composition-series-extend butlast.is-composition-series simple\( \mathcal{G} \) PnModInt Intnorm\( \mathcal{G} \) Pn unfolding \( \mathcal{G} \) Pn-def \( \Sigma \) PmInt\( \mathcal{G} \) Pn-def by auto

— Prove equality of those composition series.

have mset (map group.iso-class \( \mathcal{G} \) quotients)

= mset (map group.iso-class ((butlast comp\( \mathcal{G} \).quotients) @ [last comp\( \mathcal{G} \).quotients])) using quotes\( \mathcal{G} \) notemtpy by simp

also have \( \ldots \) = mset (map group.iso-class (butlast.quotients @ [\( \mathcal{G} \) Mod \( \mathcal{G} \) ! (\( n - 1 \))]) using comp\( \mathcal{G} \).quotients-butlast comp\( \mathcal{G} \).last-quotient length unfolding n-def \( \mathcal{G} \) Pn-def by auto

also have \( \ldots \) = mset (map group.iso-class ((butlast \( \mathcal{L} \).quotients) @ [last \( \mathcal{L} \).quotients])) + {# group.iso-class (\( \mathcal{G} \) Mod \( \mathcal{G} \) ! (\( n - 1 \))) #} using multisets\( \mathcal{G} \) butlast\( \mathcal{L} \) quotes\( \mathcal{G} \) notemtpy by simp

also have \( \ldots \) = mset (map group.iso-class (butlast.quotients @ [\( \mathcal{G} \) Pn Mod \( \mathcal{H} \) ! (\( m - 1 \)) \( \cap \) \( \mathcal{G} \) ! (\( n - 1 \))])) + {# group.iso-class (\( \mathcal{G} \) Mod \( \mathcal{G} \) ! (\( n - 1 \))) #}

56
As a corollary, we see that the composition series of a fixed group all have the same length.

**corollary (in jordan-hoelder)** jordan-hoelder-size:

shows length $\mathcal{G} = length \mathfrak{H}$

**proof**

*have* length $\mathcal{G} = length \text{comp}\mathcal{G}.\text{quotients} + 1$ by (metis comp$\mathcal{G}.\text{quotients-length})

*also have* ... = length (map group.iso-class comp$\mathcal{G}.\text{quotients}) + 1 by (metis length-map)

*also have* ... = size (mset (map group.iso-class comp$\mathcal{G}.\text{quotients})) + 1 by (metis size-mset)

*using* jordan-hoelder-multisets is-group finite is-composition-series comp$\mathfrak{H}.\text{is-composition-series} by metis

*also have* ... = length (map group.iso-class comp$\mathfrak{H}.\text{quotients}) + 1 by (metis size-mset)
also have \ldots = \text{length compf}.\text{quotients} + 1 \text{ by } (\text{metis length-map})
also have \ldots = \text{length f} \text{ by } (\text{metis compf}.\text{quotients-length})
finally show \text{thesis}.
qed
end

References