

The Jordan-Hölder Theorem

Jakob von Raumer

December 7, 2022

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

```

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
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

```

moreover have $(\lambda x. x) \text{ ' carrier } G = \text{carrier } G$ **by simp**
moreover have $\text{kernel } G \ G \ (\lambda x. x) = \{1\}$ **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 $(\lambda X. \text{the-elm } ((\lambda x. 1) \text{ ' } X)) \in \text{iso } (G \text{ Mod } (\text{carrier } G)) \ (G \ (\text{carrier } := \{1\}))$

proof –

have *group* $(G \ (\text{carrier } := \{1\}))$ **by** (*metis subgroup-imp-group triv-subgroup*)
hence *group-hom* $G \ (G \ (\text{carrier } := \{1\})) \ (\lambda x. 1)$ **unfolding** *group-hom-def group-hom-axioms-def hom-def* **using** *is-group* **by auto**
moreover have $(\lambda x. 1) \text{ ' carrier } G = \text{carrier } (G \ (\text{carrier } := \{1\}))$ **by auto**
moreover have $\text{kernel } G \ (G \ (\text{carrier } := \{1\})) \ (\lambda x. 1) = \text{carrier } G$ **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 $S::'a \text{ set}$
assumes *subgrpS:subgroup S G*

context *second-isomorphism-grp*
begin

interpretation *groupS: group G(carrier := S)*
using *subgrpS* **by** (*metis subgroup-imp-group*)

lemma *normal-subgrp-intersection-normal*:

shows $S \cap H \triangleleft (G \ (\text{carrier } := S))$
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* $(S \cap H) \ (G \ (\text{carrier } := S))$ **using** *is-group subgroup.subgroup-of-subset subgrpS* **by metis**
next
fix $g \ h$
assume $g: g \in S$ **and** $hH: h \in H$ **and** $hS: h \in S$ {
from $g \ hH \ \text{subgrpS}$ **show** $g \otimes h \otimes \text{inv } G \ (\text{carrier } := S) \ g \in H$ **by** (*metis inv-op-closed2 subgroup.mem-carrier m-inv-consistent*)

```

} {
  from g hS subgrpS show g ⊗ h ⊗ invG(carrier := S) g ∈ S by (metis sub-
group.m-closed subgroup.m-inv-closed m-inv-consistent)
}
qed

```

lemma *normal-set-mult-subgroup*:

shows *subgroup* (H <#> S) G

proof(rule *subgroupI*)

show H <#> S ⊆ carrier G **by** (metis *setmult-subset-G subgroup.subset subgrpS subset*)

next

have 1 ∈ H 1 ∈ S **using** *is-subgroup subgrpS subgroup.one-closed* **by** *auto*

hence 1 ⊗ 1 ∈ H <#> S **unfolding** *set-mult-def* **by** *blast*

thus H <#> S ≠ {} **by** *auto*

next

fix g

assume g : g ∈ H <#> S

then obtain h s **where** h : h ∈ H **and** s : s ∈ S **and** ghs : g = h ⊗ s **unfolding** *set-mult-def* **by** *auto*

hence s ∈ carrier G **by** (metis *subgroup.mem-carrier subgrpS*)

with h ghs **obtain** h' **where** h' : h' ∈ H **and** g = s ⊗ h' **using** *coset-eq* **unfolding** *r-coset-def l-coset-def* **by** *auto*

with s **have** inv g = (inv h') ⊗ (inv s) **by** (metis *inv-mult-group mem-carrier subgroup.mem-carrier subgrpS*)

moreover from h' s subgrpS **have** inv h' ∈ H inv s ∈ S **using** *subgroup.m-inv-closed m-inv-closed* **by** *auto*

ultimately show inv g ∈ H <#> S **unfolding** *set-mult-def* **by** *auto*

next

fix g g'

assume g : g ∈ H <#> S **and** h : g' ∈ H <#> S

then obtain h h' s s' **where** hh'ss' : h ∈ H h' ∈ H s ∈ S s' ∈ S **and** g = h ⊗ s **and** g' = h' ⊗ s' **unfolding** *set-mult-def* **by** *auto*

hence g ⊗ g' = (h ⊗ s) ⊗ (h' ⊗ s') **by** *metis*

also from hh'ss' **have** inG : h ∈ carrier G h' ∈ carrier G s ∈ carrier G s' ∈ carrier G **using** *subgrpS mem-carrier subgroup.mem-carrier* **by** *force+*

hence (h ⊗ s) ⊗ (h' ⊗ s') = h ⊗ (s ⊗ h') ⊗ s' **using** *m-assoc* **by** *auto*

also from hh'ss' inG **obtain** h'' **where** h'' : h'' ∈ H **and** s ⊗ h' = h'' ⊗ s **using** *coset-eq* **unfolding** *r-coset-def l-coset-def* **by** *fastforce*

hence h ⊗ (s ⊗ h') ⊗ s' = h ⊗ (h'' ⊗ s) ⊗ s' **by** *simp*

also from h'' inG **have** ... = (h ⊗ h'') ⊗ (s ⊗ s') **using** *m-assoc mem-carrier* **by** *auto*

finally have g ⊗ g' = h ⊗ h'' ⊗ (s ⊗ s').

moreover with h'' hh'ss' **have** ... ∈ H <#> S **unfolding** *set-mult-def* **using** *subgrpS subgroup.m-closed* **by** *fastforce*

ultimately show g ⊗ g' ∈ H <#> S **by** *simp*

qed

lemma *oneH* : 1 ∈ H **by** (metis *is-subgroup subgroup.one-closed*)

```

lemma H-contained-in-set-mult:
  shows  $H \subseteq H \langle \# \rangle S$ 
proof auto
  have  $1 \in S$  by (metis subgroup.one-closed subgrpS)
  fix  $x$ 
  assume  $x \in H$ 
  with  $\langle 1 \in S \rangle$  have  $x \otimes 1 \in H \langle \# \rangle S$  unfolding set-mult-def by force
  with  $x$  show  $x \in H \langle \# \rangle S$  by (metis mem-carrier r-one)
qed

lemma S-contained-in-set-mult:
  shows  $S \subseteq H \langle \# \rangle S$ 
proof auto
  fix  $s$ 
  assume  $s \in S$ 
  with oneH have  $1 \otimes s \in H \langle \# \rangle S$  unfolding set-mult-def by force
  with  $s$  show  $s \in H \langle \# \rangle S$  using subgrpS subgroup.mem-carrier l-one by force
qed

lemma normal-intersection-hom:
  shows group-hom ( $G \langle \text{carrier} := S \rangle$ ) (( $G \langle \text{carrier} := H \langle \# \rangle S \rangle$ ) Mod  $H$ ) ( $\lambda g.$ 
 $H \langle \# \rangle g$ )
proof (auto del: equalityI simp: group-hom-def group-hom-axioms-def hom-def groupS.is-group)
  have gr:group ( $G \langle \text{carrier} := H \langle \# \rangle S \rangle$ ) by (metis normal-set-mult-subgroup
subgroup-imp-group)
  moreover have  $H \subseteq H \langle \# \rangle S$  by (rule H-contained-in-set-mult)
  moreover have subgroup ( $H \langle \# \rangle S$ )  $G$  by (metis normal-set-mult-subgroup)
  ultimately have  $H \triangleleft (G \langle \text{carrier} := H \langle \# \rangle S \rangle)$  using normal-restrict-supergroup
by (metis inv-op-closed2 is-subgroup normal-inv-iff)
  with gr show group (( $G \langle \text{carrier} := H \langle \# \rangle S \rangle$ ) Mod  $H$ ) by (metis normal.factorgroup-is-group)
next
  fix  $g$ 
  assume  $g \in S$ 
  with subgrpS have  $1 \otimes g \in H \langle \# \rangle S$  unfolding set-mult-def by fastforce
  with  $g$  have  $g \in H \langle \# \rangle S$  by (metis l-one subgroup.mem-carrier subgrpS)
  thus  $H \langle \# \rangle g \in \text{carrier} ((G \langle \text{carrier} := H \langle \# \rangle S \rangle) \text{Mod } H)$  unfolding Fact-Group-def RCOSETS-def r-coset-def by auto
next
  show  $\bigwedge x y. [x \in S; y \in S] \implies H \langle \# \rangle x \otimes y = H \langle \# \rangle x \langle \# \rangle (H \langle \# \rangle y)$ 
    using normal.rcos-sum normal-axioms subgroup.mem-carrier subgrpS by fastforce
qed

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

```

```

have kernel (G⟦carrier := S⟧) ((G⟦carrier := H <#> S⟧) Mod H) (λg. H #>
g)
      = {g ∈ S. H #> g = 1(G⟦carrier := H <#> S⟧) Mod H} unfolding
kernel-def by auto
also have ... = {g ∈ S. H #> g = H} unfolding FactGroup-def by auto
also have ... = {g ∈ S. g ∈ H} by (metis coset-eq is-subgroup lcoset-join2 rcos-self
subgroup.mem-carrier subgrpS)
also have ... = H ∩ S by auto
finally show ?thesis.
qed

```

lemma *normal-intersection-hom-surj*:

```

shows (λg. H #> g) ‘carrier (G⟦carrier := S⟧) = carrier ((G⟦carrier := H
<#> S⟧) Mod H)

```

proof *auto*

```

fix g
assume g ∈ S
hence g ∈ H <#> S using S-contained-in-set-mult by auto
thus H #> g ∈ carrier ((G⟦carrier := H <#> S⟧) Mod H) unfolding Fact-
Group-def RCOSETS-def r-coset-def by auto
next
fix x
assume x ∈ carrier (G⟦carrier := H <#> S⟧) Mod H)
then obtain h s where h:h ∈ H and s:s ∈ S and x = H #> (h ⊗ 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 sub-
group.mem-carrier subgrpS subset)
also have ... = H #> s by (metis h is-group rcos-const)
finally have x = H #> s.
with s show x ∈ (#>) H ‘ S by simp
qed

```

Finally we can prove the actual isomorphism theorem:

theorem *normal-intersection-quotient-isom*:

```

shows (λX. the-elem ((λg. H #> g) ‘ X)) ∈ iso ((G⟦carrier := S⟧) Mod (H ∩
S)) (((G⟦carrier := H <#> S⟧) Mod H)

```

```

using normal-intersection-hom-kernel[symmetric] normal-intersection-hom normal-intersection-hom-surj
by (metis group-hom.FactGroup-iso-set)

```

end

end

theory *SubgroupsAndNormalSubgroups*

imports

Secondary-Sylow.SndSylow

SndIsomorphismGrp

HOL-Algebra.Coset

begin

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 $\text{order:order } G = 1$

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 *finite:finite* ($\text{carrier } G$)

shows $0 < \text{order } G$

proof –

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

qed

lemma *iso-order-closed*:

assumes $\varphi \in \text{iso } G H$

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 *subgroup* U ($G \setminus \text{carrier} := H$)

shows $U \subseteq H$

using *assms subgroup.subset* **by** *force*

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

assumes *group* G

assumes *subgroup* U ($G \setminus \text{carrier} := H$)

shows *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 ⊗ y = x ⊗G (carrier := H) y by simp
assume x ∈ U y ∈ U
with assms a show x ⊗ y ∈ U by (metis subgroup.m-closed)
next
have 1G (carrier := H) = 1 by simp
with assms show 1 ∈ U by (metis subgroup.one-closed)
next
have subgroup H G..
fix x
assume x ∈ U
with assms(2) have invG (carrier := H) x ∈ U by (rule subgroup.m-inv-closed)
moreover from assms ⟨x ∈ U⟩ have x ∈ H by (metis in-mono subgroup-of-restricted-group)
with assms(1) ⟨subgroup H G⟩ have invG (carrier := H) x = inv x by (rule
group.m-inv-consistent)
ultimately show inv x ∈ U by simp
qed

```

Being a subgroup is preserved by surjective homomorphisms

```

lemma (in subgroup) surj-hom-subgroup:
  assumes φ:group-hom G F φ
  assumes φsurj:φ ‘ (carrier G) = carrier F
  shows subgroup (φ ‘ H) F
proof
  from φsurj show img-subset:φ ‘ H ⊆ carrier F unfolding iso-def bij-betw-def
by auto
next
fix f f'
assume h:f ∈ φ ‘ H and h':f' ∈ φ ‘ H
with φsurj obtain g g' where g:g ∈ H f = φ g and g':g' ∈ H f' = φ g' by auto
hence g ⊗G g' ∈ H by (metis m-closed)
hence φ (g ⊗G g') ∈ φ ‘ H by simp
with g g' φ show f ⊗F f' ∈ φ ‘ H using group-hom.hom-mult by fastforce
next
have φ 1 ∈ φ ‘ H by auto
with φ show 1F ∈ φ ‘ H by (metis group-hom.hom-one)
next
fix f
assume f:f ∈ φ ‘ H
then obtain g where g:g ∈ H f = φ g by auto
hence inv g ∈ H by auto
hence φ (inv g) ∈ φ ‘ H by auto
with φ g subset show invF f ∈ φ ‘ H using group-hom.hom-inv by fastforce
qed

```

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

```

lemma iso-subgroup:
  assumes groups:group G group F

```

```

assumes  $HG: \text{subgroup } H \ G$ 
assumes  $\varphi: \varphi \in \text{iso } G \ F$ 
shows  $\text{subgroup } (\varphi \text{ ` } H) \ F$ 
proof –
  from  $\text{groups } \varphi$  have  $\text{group-hom } G \ F \ \varphi$  unfolding  $\text{group-hom-def group-hom-axioms-def}$ 
 $\text{iso-def}$  by  $\text{auto}$ 
  moreover from  $\varphi$  have  $\varphi \text{ ` } (\text{carrier } G) = \text{carrier } F$  unfolding  $\text{iso-def bij-betw-def}$ 
by  $\text{simp}$ 
  moreover note  $HG$ 
  ultimately show  $?thesis$  by  $(\text{metis subgroup.surj-hom-subgroup})$ 
qed

```

An isomorphism restricts to an isomorphism of subgroups.

lemma iso-restrict :

```

assumes  $\text{groups:group } G \ \text{group } F$ 
assumes  $HG: \text{subgroup } H \ G$ 
assumes  $\varphi: \varphi \in \text{iso } G \ F$ 
shows  $(\text{restrict } \varphi \ H) \in \text{iso } (G \ \backslash \ \text{carrier } := \ H) \ (F \ \backslash \ \text{carrier } := \ \varphi \text{ ` } H)$ 
unfolding  $\text{iso-def hom-def bij-betw-def inj-on-def}$ 
proof  $\text{auto}$ 
  fix  $g \ h$ 
  assume  $g \in H \ h \in H$ 
  hence  $g \in \text{carrier } G \ h \in \text{carrier } G$  by  $(\text{metis } HG \ \text{subgroup.mem-carrier})+$ 
  thus  $\varphi \ (g \otimes_G \ h) = \varphi \ g \otimes_F \ \varphi \ h$  using  $\varphi$  unfolding  $\text{iso-def hom-def}$  by  $\text{auto}$ 
next
  fix  $g \ h$ 
  assume  $g \in H \ h \in H \ g \otimes_G \ h \notin H$ 
  hence  $\text{False}$  using  $HG$  unfolding  $\text{subgroup-def}$  by  $\text{auto}$ 
  thus  $\text{undefined} = \varphi \ g \otimes_F \ \varphi \ h$  by  $\text{auto}$ 
next
  fix  $g \ h$ 
  assume  $g: g \in H$  and  $h: h \in H$  and  $\text{eq}: \varphi \ g = \varphi \ h$ 
  hence  $g \in \text{carrier } G \ h \in \text{carrier } G$  by  $(\text{metis } HG \ \text{subgroup.mem-carrier})+$ 
  with  $\text{eq}$  show  $g = h$  using  $\varphi$  unfolding  $\text{iso-def bij-betw-def inj-on-def}$  by  $\text{auto}$ 
qed

```

The intersection of two subgroups is, again, a subgroup

lemma $(\text{in } \text{group}) \ \text{subgroup-intersect}$:

```

assumes  $\text{subgroup } H \ G$ 
assumes  $\text{subgroup } H' \ G$ 
shows  $\text{subgroup } (H \cap H') \ G$ 
using  $\text{assms}$  unfolding  $\text{subgroup-def}$  by  $\text{auto}$ 

```

4 Facts about Normal Subgroups

lemma $(\text{in } \text{normal}) \ \text{is-normal}$:

```

shows  $H \triangleleft G$ 
by  $(\text{metis coset-eq is-subgroup normalI})$ 

```

Being a normal subgroup is preserved by surjective homomorphisms.

lemma (in *normal*) *surj-hom-normal-subgroup*:

assumes φ :group-hom $G F$ φ
 assumes φ surj: φ '(carrier G) = carrier F
 shows $(\varphi$ ' H) $\triangleleft F$

proof (rule *group.normalI*)

from φ show group F **unfolding** *group-hom-def group-hom-axioms-def* **by** *simp*
next

from φ φ surj show subgroup $(\varphi$ ' H) F **by** (rule *surj-hom-subgroup*)

next

show $\forall x \in \text{carrier } F. \varphi$ ' H $\#>_F x = x <\#_F \varphi$ ' H

proof

fix f

assume $f:f \in \text{carrier } F$

with φ surj **obtain** g where $g:g \in \text{carrier } G f = \varphi g$ **by** *auto*

hence φ ' H $\#>_F f = \varphi$ ' H $\#>_F \varphi g$ **by** *simp*

also have ... = $(\lambda x. (\varphi x) \otimes_F (\varphi g))$ ' H **unfolding** *r-coset-def image-def* **by** *auto*

also have ... = $(\lambda x. \varphi (x \otimes g))$ ' H **using** *subset g φ group-hom.hom-mult* **unfolding** *image-def* **by** *fastforce*

also have ... = φ ' $(H \#> g)$ **using** φ **unfolding** *r-coset-def* **by** *auto*

also have ... = φ ' $(g <\# H)$ **by** (*metis coset-eq g(1)*)

also have ... = $(\lambda x. \varphi (g \otimes x))$ ' H **using** φ **unfolding** *l-coset-def* **by** *auto*

also have ... = $(\lambda x. (\varphi g) \otimes_F (\varphi x))$ ' H **using** *subset g φ group-hom.hom-mult* **by** *fastforce*

also have ... = $\varphi g <\#_F \varphi$ ' H **unfolding** *l-coset-def image-def* **by** *auto*

also have ... = $f <\#_F \varphi$ ' H **using** g **by** *simp*

finally show φ ' H $\#>_F f = f <\#_F \varphi$ ' H .

qed

qed

Being a normal subgroup is preserved by group isomorphisms.

lemma *iso-normal-subgroup*:

assumes *groups*:group G group F

assumes $HG:H \triangleleft G$

assumes $\varphi:\varphi \in \text{iso } G F$

shows $(\varphi$ ' H) $\triangleleft F$

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* **using** *normal.surj-hom-normal-subgroup* **by** *metis*

qed

The trivial subgroup is a subgroup:

lemma (in *group*) *triv-subgroup*:

shows subgroup $\{1\} G$

unfolding *subgroup-def* **by** *auto*

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

lemma (*in group*) *card-rcosets-triv*:
assumes *finite* (*carrier G*)
shows $\text{card } (\text{rcosets } \{1\}) = \text{order } G$
proof –
have *subgroup* $\{1\} \ G$ **by** (*rule triv-subgroup*)
with *assms* **have** $\text{card } (\text{rcosets } \{1\}) * \text{card } \{1\} = \text{order } G$
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 $M \triangleleft G$ **and** $N \triangleleft G$
shows $M \cap N \triangleleft G$
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*:
 $\llbracket H \subseteq \text{carrier } G; K \subseteq \text{carrier } G; x \in \text{carrier } G \rrbracket$
 $\implies (x \langle \# \rangle H) \langle \# \rangle K = x \langle \# \rangle (H \langle \# \rangle K)$
by (*force simp add: l-coset-def set-mult-def m-assoc*)

lemma (*in group*) *normal-subgroup-set-mult-closed*:
assumes $M \triangleleft G$ **and** $N \triangleleft G$
shows $M \langle \# \rangle N \triangleleft G$
proof (*rule normalI*)
from *assms* **show** *subgroup* $(M \langle \# \rangle N) \ G$
using *second-isomorphism-grp.normal-set-mult-subgroup normal-imp-subgroup*
unfolding *second-isomorphism-grp-def second-isomorphism-grp-axioms-def* **by**
force
next
show $\forall x \in \text{carrier } G. M \langle \# \rangle N \# \rangle x = x \langle \# \rangle (M \langle \# \rangle N)$
proof
fix x
assume $x \in \text{carrier } G$
have $M \langle \# \rangle N \# \rangle x = M \langle \# \rangle (N \# \rangle x)$ **by** (*metis assms(1,2) normal-inv-iff setmult-rcos-assoc subgroup.subset x*)
also have $\dots = M \langle \# \rangle (x \langle \# \rangle N)$ **by** (*metis assms(2) normal.coset-eq x*)
also have $\dots = (M \# \rangle x) \langle \# \rangle N$ **by** (*metis assms(1,2) normal-imp-subgroup rcos-assoc-lcos subgroup.subset x*)
also have $\dots = (x \langle \# \rangle M) \langle \# \rangle N$ **by** (*metis assms(1) normal.coset-eq x*)
also have $\dots = x \langle \# \rangle (M \langle \# \rangle N)$ **by** (*metis assms(1,2) normal-imp-subgroup setmult-lcos-assoc subgroup.subset x*)
finally show $M \langle \# \rangle N \# \rangle x = x \langle \# \rangle (M \langle \# \rangle N)$.
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 $x \in \text{carrier } (G(\text{carrier} := H)) y \in \text{carrier } (G(\text{carrier} := H))$ **by** *auto*

with *assms(2)* **show** $x \otimes y \in H$ **using** *assms(2)* *group.is-monoid monoid.m-closed*

by *fastforce*

next

show $1 \in H$ **using** *assms(2)* *group.is-monoid monoid.one-closed* **by** *fastforce*

next

fix x

assume $x \in H$

hence $x \in \text{carrier } (G(\text{carrier} := H))$ **by** *auto*

hence $\text{inv}_{G(\text{carrier} := H)} x \in \text{carrier } (G(\text{carrier} := H))$ **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.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** *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 $N \triangleleft G$ **and** $N \subseteq H$ **and** *subgroup* $H G$

shows *subgroup* (*rcosets* $_{G(\text{carrier} := H)} N$) ($G \text{ Mod } N$)

proof –

interpret $G \text{ Mod } N$: *group* $G \text{ Mod } N$ **using** *assms(1)* **by** (*rule normal.factorgroup-is-group*)

have $N \triangleleft G(\text{carrier} := H)$ **using** *assms* **by** (*metis normal-restrict-supergroup*)

hence grpHN : *group* ($G(\text{carrier} := H) \text{ Mod } N$) **by** (*rule normal.factorgroup-is-group*)

have $\langle \# \rangle_{G(\text{carrier} := H)} = (\lambda U K. (\bigcup h \in U. \bigcup k \in K. \{h \otimes_{G(\text{carrier} := H)} k\}))$ **using** *set-mult-def* **by** *metis*

moreover **have** $\dots = (\lambda U K. (\bigcup h \in U. \bigcup k \in K. \{h \otimes_G k\}))$ **by** *auto*

moreover **have** $\langle \# \rangle = (\lambda U K. (\bigcup h \in U. \bigcup k \in K. \{h \otimes k\}))$ **using** *set-mult-def*

by *metis*

ultimately **have** $\langle \# \rangle_{G(\text{carrier} := H)} = \langle \# \rangle_G$ **by** *simp*

with *grpHN* **have** *group* ((*G Mod N*)(*carrier := (rcosets*_{*G*}(*carrier := H*) *N*)))
unfolding *FactGroup-def* **by** *auto*
moreover **have** *rcosets*_{*G*}(*carrier := H*) *N* \subseteq *carrier* (*G Mod N*) **unfolding**
FactGroup-def 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*}(*carrier := H*) *N*) \triangleleft (*G Mod N*)

proof –

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

show *?thesis*

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

from *assms* **show** *subgroup* (*rcosets*_{*G*}(*carrier := H*) *N*) (*G Mod N*) **using**
normal-imp-subgroup normal-subgroup-factorize **by** *force*

next

fix *U V*

assume *U:U* \in *carrier* (*G Mod N*) **and** *V:V* \in *rcosets*_{*G*}(*carrier := H*) *N*

then obtain *g* **where** *g:g* \in *carrier* *G* *U = N* $\#>$ *g* **unfolding** *FactGroup-def*
RCOSETS-def **by** *auto*

from *V* **obtain** *h* **where** *h:h* \in *H* *V = N* $\#>$ *h* **unfolding** *FactGroup-def*
RCOSETS-def r-coset-def **by** *auto*

hence *hG:h* \in *carrier* *G* **using** *HG normal-imp-subgroup subgroup.mem-carrier*
by *force*

hence *ghG:g* \otimes *h* \in *carrier* *G* **using** *g m-closed* **by** *auto*

from *g h* **have** *g* \otimes *h* \otimes *inv g* \in *H* **using** *HG normal-inv-iff* **by** *auto*

moreover **have** *U* $\langle \# \rangle$ *V* $\langle \# \rangle$ *inv*_{*G Mod N*} *U = N* $\#>$ (*g* \otimes *h* \otimes *inv g*)

proof –

from *g U* **have** *inv*_{*G Mod N*} *U = N* $\#>$ *inv g* **using** *NG normal.inv-FactGroup*
normal.rcos-inv **by** *fastforce*

hence *U* $\langle \# \rangle$ *V* $\langle \# \rangle$ *inv*_{*G Mod N*} *U = (N* $\#>$ *g)* $\langle \# \rangle$ (*N* $\#>$ *h)* $\langle \# \rangle$
(*N* $\#>$ *inv g)* **using** *g h* **by** *simp*

also **have** $\dots = N$ $\#>$ (*g* \otimes *h)* $\langle \# \rangle$ (*N* $\#>$ *inv g)* **using** *g hG NG*
normal.rcos-sum **by** *force*

also **have** $\dots = N$ $\#>$ (*g* \otimes *h* \otimes *inv g)* **using** *g inv-closed ghG NG*
normal.rcos-sum **by** *force*

finally show *?thesis* .

qed

ultimately show *U* $\langle \# \rangle$ *V* $\langle \# \rangle$ *inv*_{*G Mod N*} *U* \in *rcosets*_{*G*}(*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* (carrier G)
shows (carrier ($G \text{ Mod } H$) = $\{1_{G \text{ Mod } H}\}$) = ($H = \text{carrier } G$)
proof
assume carrier ($G \text{ Mod } H$) = $\{1_{G \text{ Mod } H}\}$
moreover with *assms lagrange* **have** order ($G \text{ Mod } H$) * card $H = \text{order } G$
unfolding *FactGroup-def order-def* **using** *is-subgroup* **by force**
ultimately have card $H = \text{order } G$ **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** *ordergt0:order* $G > 0$ **unfolding** *order-def* **by** (*metis subgroup.finite-imp-card-positive subgroup-self*)
assume $H = \text{carrier } G$
hence card $H = \text{order } G$ **unfolding** *order-def* **by simp**
with *assms is-subgroup lagrange* **have** card (*rcosets* H) * order $G = \text{order } G$ **by** *metis*
with *ordergt0* **have** card (*rcosets* H) = 1 **by** (*metis mult-eq-self-implies-10 mult.commute neq0-conv*)
hence order ($G \text{ Mod } H$) = 1 **unfolding** *order-def FactGroup-def* **by auto**
thus carrier ($G \text{ Mod } H$) = $\{1_{G \text{ Mod } H}\}$ **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 ($G \text{ Mod } H$)
shows $(\bigcup A) = \{x \in \text{carrier } G. H \#> x \in A\}$
proof
show $\bigcup A \subseteq \{x \in \text{carrier } G. H \#> x \in A\}$
proof
fix x
assume $x \in \bigcup A$
then obtain a **where** $a \in A$ $x \in a$ **by auto**
with *assms* **have** $x \in \text{carrier } G$ **using** *subgroup.subset* **unfolding** *FactGroup-def RCOSETS-def r-coset-def* **by force**
from *assms* a **obtain** y **where** $y \in \text{carrier } G$ $a = H \#> y$ **using** *subgroup.subset* **unfolding** *FactGroup-def RCOSETS-def* **by force**
with a **have** $x \in H \#> y$ **by simp**
hence $H \#> y = H \#> x$ **using** *y is-subgroup repr-independence* **by auto**
with $y^{(2)}$ $a^{(1)}$ **have** $H \#> x \in A$ **by auto**

```

    with  $xx$  show  $x \in \{x \in \text{carrier } G. H \#> x \in A\}$  by simp
  qed
next
show  $\{x \in \text{carrier } G. H \#> x \in A\} \subseteq \bigcup A$ 
proof
  fix  $x$ 
  assume  $x:x \in \{x \in \text{carrier } G. H \#> x \in A\}$ 
  hence  $xx:x \in \text{carrier } G H \#> x \in A$  by auto
  moreover have  $x \in H \#> x$  by (metis is-subgroup rcos-self  $xx(1)$ )
  ultimately show  $x \in \bigcup A$  by auto
qed
qed

lemma (in normal) factgroup-subgroup-union-subgroup:
  assumes subgroup  $A (G \text{ Mod } H)$ 
  shows subgroup  $(\bigcup A) G$ 
proof -
  have subgroup  $\{x \in \text{carrier } G. H \#> x \in A\} G$ 
  proof
    show  $\{x \in \text{carrier } G. H \#> x \in A\} \subseteq \text{carrier } G$  by auto
  next
    fix  $x y$ 
    assume  $x \in \{x \in \text{carrier } G. H \#> x \in A\}$  and  $y \in \{x \in \text{carrier } G. H \#> x \in A\}$ 
    hence  $x:x \in \text{carrier } G H \#> x \in A$  and  $y:y \in \text{carrier } G H \#> y \in A$  by auto
    hence  $xyG:x \otimes y \in \text{carrier } G$  by (metis m-closed)
    from  $assms x y$  have  $(H \#> x) <\#> (H \#> y) \in A$  using subgroup.m-closed
  unfolding FactGroup-def by fastforce
  hence  $H \#> (x \otimes y) \in A$  by (metis rcos-sum  $x(1) y(1)$ )
  with  $xyG$  show  $x \otimes y \in \{x \in \text{carrier } G. H \#> x \in A\}$  by simp
  next
    have  $H \#> 1 \in A$  using  $assms$  subgroup.one-closed unfolding FactGroup-def
  by (metis coset-mult-one monoid.select-conv(2) subset)
  with  $assms$  one-closed show  $1 \in \{x \in \text{carrier } G. H \#> x \in A\}$  by simp
  next
    fix  $x$ 
    assume  $x \in \{x \in \text{carrier } G. H \#> x \in A\}$ 
    hence  $x:x \in \text{carrier } G H \#> x \in A$  by auto
    hence  $invx:inv x \in \text{carrier } G$  using inv-closed by simp
    from  $assms x$  have  $set-inv (H \#> x) \in A$  using subgroup.m-inv-closed by
  (metis inv-FactGroup subgroup.mem-carrier)
    hence  $H \#> (inv x) \in A$  by (metis rcos-inv  $x(1)$ )
    with  $invx$  show  $inv x \in \{x \in \text{carrier } G. H \#> x \in A\}$  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  $A (G \text{ Mod } H)$  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  $x' \in \{x \in \text{carrier } G. H \#> x \in A\} \ y = x' \otimes x$ 
  unfolding r-coset-def by auto
    hence  $x': x' \in \text{carrier } G \ H \#> x' \in A$  by auto
    from  $x(1)$  have  $Hx: H \#> x \in \text{carrier } (G \text{ Mod } H)$  unfolding FactGroup-def RCOSETS-def by force
    with  $x'$  have  $(\text{inv}_{G \text{ Mod } H} (H \#> x)) \otimes_{G \text{ Mod } H} (H \#> x') \otimes_{G \text{ Mod } H} (H \#> x) \in A$  using Anormal.inv-op-closed1 by auto
    hence  $(\text{set-inv } (H \#> x)) \langle \# \rangle (H \#> x') \langle \# \rangle (H \#> x) \in A$  using inv-FactGroup Hx unfolding FactGroup-def by auto
    hence  $(H \#> (\text{inv } x)) \langle \# \rangle (H \#> x') \langle \# \rangle (H \#> x) \in A$  using  $x(1)$  by (metis rcos-inv)
    hence  $(H \#> (\text{inv } x \otimes x')) \langle \# \rangle (H \#> x) \in A$  by (metis inv-closed rcos-sum x'(1) x(1))
    hence  $H \#> (\text{inv } x \otimes x' \otimes x) \in A$  by (metis inv-closed m-closed rcos-sum x'(1) x(1))
    moreover have  $\text{inv } x \otimes x' \otimes x \in \text{carrier } G$  using  $x \ x'$  by (metis inv-closed m-closed)
    ultimately have  $\text{inv } x \otimes x' \otimes x \in \{x \in \text{carrier } G. H \#> x \in A\}$  by auto
    hence  $x\text{coset}: x \otimes (\text{inv } x \otimes x' \otimes x) \in x \langle \# \rangle \{x \in \text{carrier } G. H \#> x \in A\}$ 
  unfolding l-coset-def using  $x(1)$  by auto
    have  $x \otimes (\text{inv } x \otimes x' \otimes x) = (x \otimes \text{inv } x) \otimes x' \otimes x$  by (metis Units-eq Units-inv-Units m-assoc m-closed x'(1) x(1))
    also have  $\dots = x' \otimes x$  by (metis l-one r-inv x'(1) x(1))
    also have  $\dots = y$  by (metis ‹y = x' ‹x›)
    finally have  $x \otimes (\text{inv } x \otimes x' \otimes x) = y$ .
    with xcoset show  $y \in x \langle \# \rangle \{x \in \text{carrier } G. H \#> x \in A\}$  by auto
  next
    interpret Anormal: normal  $A (G \text{ Mod } H)$  using assms by simp
    fix  $x y$ 
    assume  $x: x \in \text{carrier } G \ y \in x \langle \# \rangle \{x \in \text{carrier } G. H \#> x \in A\}$ 
    then obtain  $x'$  where  $x' \in \{x \in \text{carrier } G. H \#> x \in A\} \ y = x \otimes x'$ 
  unfolding l-coset-def by auto
    hence  $x': x' \in \text{carrier } G \ H \#> x' \in A$  by auto
    from  $x(1)$  have  $\text{inv } x: \text{inv } x \in \text{carrier } G$  by (rule inv-closed)
    hence  $H\text{inv } x: H \#> (\text{inv } x) \in \text{carrier } (G \text{ Mod } H)$  unfolding FactGroup-def RCOSETS-def by force

```

with x' **have** $(\text{inv}_G \text{Mod } H (H \#> \text{inv } x)) \otimes_G \text{Mod } H (H \#> x') \otimes_G \text{Mod } H (H \#> \text{inv } x) \in A$ **using** $\text{inv } x$ *Anormal.inv-op-closed1* **by** *auto*
hence $(\text{set-inv } (H \#> \text{inv } x)) <\#\> (H \#> x') <\#\> (H \#> \text{inv } x) \in A$
using *inv-FactGroup Hinvx unfolding FactGroup-def* **by** *auto*
hence $(H \#> \text{inv } (\text{inv } x)) <\#\> (H \#> x') <\#\> (H \#> \text{inv } x) \in A$ **using** $\text{inv } x$ **by** *(metis rcos-inv)*
hence $(H \#> x) <\#\> (H \#> x') <\#\> (H \#> \text{inv } x) \in A$ **by** *(metis inv-inv x(1))*
hence $(H \#> (x \otimes x')) <\#\> (H \#> \text{inv } x) \in A$ **by** *(metis rcos-sum x'(1) x(1))*
hence $H \#> (x \otimes x' \otimes \text{inv } x) \in A$ **by** *(metis inv-closed m-closed rcos-sum x'(1) x(1))*
moreover **have** $x \otimes x' \otimes \text{inv } x \in \text{carrier } G$ **using** x x' **by** *(metis inv-closed m-closed)*
ultimately **have** $x \otimes x' \otimes \text{inv } x \in \{x \in \text{carrier } G. H \#> x \in A\}$ **by** *auto*
hence $\text{xcoset}:(x \otimes x' \otimes \text{inv } x) \otimes x \in \{x \in \text{carrier } G. H \#> x \in A\} \#> x$
unfolding *r-coset-def* **using** $\text{inv } x$ **by** *auto*
have $(x \otimes x' \otimes \text{inv } x) \otimes x = (x \otimes x') \otimes (\text{inv } x \otimes x)$ **by** *(metis Units-eq Units-inv-Units m-assoc m-closed x'(1) x(1))*
also **have** $\dots = x \otimes x'$ **using** $x(1)$ *l-inv x'(1) m-closed r-one* **by** *auto*
also **have** $\dots = y$ **by** *(metis <y = x \otimes x')*
finally **have** $x \otimes x' \otimes \text{inv } x \otimes x = y$.
with xcoset **show** $y \in \{x \in \text{carrier } G. H \#> x \in A\} \#> x$ **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 $A = \text{rcosets}_G(\text{carrier} := \bigcup A) H$

proof –

have $A = \text{rcosets}_G(\text{carrier} := \{x \in \text{carrier } G. H \#> x \in A\}) H$

proof *auto*

fix U

assume $U:U \in A$

then **obtain** x' **where** $x':x' \in \text{carrier } G$ $U = H \#> x'$ **using** assms *subgroup.subset* **unfolding** *FactGroup-def RCOSETS-def* **by** *force*

with U **have** $H \#> x' \in A$ **by** *simp*

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

next

fix U

assume $U:U \in \text{rcosets}_G(\text{carrier} := \{x \in \text{carrier } G. H \#> x \in A\}) H$

then **obtain** x' **where** $x':x' \in \{x \in \text{carrier } G. H \#> x \in A\}$ $U = H \#> x'$ **unfolding** *RCOSETS-def r-coset-def* **by** *auto*

hence $x' \in \text{carrier } G$ $H \#> x' \in A$ **by** *auto*

with x' **show** $U \in A$ **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**

flatten (*G::('a set, 'b) monoid-scheme*) *rep* = (\langle *carrier*=(*rep* ' (*carrier* *G*)),
 \langle *monoid.mult*=(λ *x y. rep* ((*the-inv-into* (*carrier* *G*) *rep* *x*) \otimes_G (*the-inv-into*
(*carrier* *G*) *rep* *y*))),
 \langle *one*=*rep* **1**_{*G*} \rangle)

lemma *flatten-set-group-hom*:

assumes *group:group* *G*

assumes *inj:inj-on rep* (*carrier* *G*)

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

unfolding *hom-def*

proof *auto*

fix *g*

assume *g: g* \in *carrier* *G*

thus *rep* *g* \in *carrier* (*flatten* *G* *rep*) **unfolding** *flatten-def* **by** *auto*

next

fix *g h*

assume *g: g* \in *carrier* *G* **and** *h: h* \in *carrier* *G*

hence *rep* *g* \in *carrier* (*flatten* *G* *rep*) *rep* *h* \in *carrier* (*flatten* *G* *rep*) **unfolding**
flatten-def **by** *auto*

hence *rep* *g* $\otimes_{\text{flatten } G \text{ rep}}$ *rep* *h*

= *rep* (*the-inv-into* (*carrier* *G*) *rep* (*rep* *g*) \otimes_G *the-inv-into* (*carrier* *G*) *rep* (*rep*
h)) **unfolding** *flatten-def* **by** *auto*

also have ... = *rep* (*g* \otimes_G *h*) **using** *inj* *g h* **by** (*metis the-inv-into-f-f*)

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

qed

lemma *flatten-set-group*:

assumes *group:group* *G*

assumes *inj:inj-on rep* (*carrier* *G*)

shows *group* (*flatten* *G* *rep*)

proof (*rule groupI*)

fix *x y*

assume *x: x* \in *carrier* (*flatten* *G* *rep*) **and** *y: y* \in *carrier* (*flatten* *G* *rep*)

define *g h*

where *g* = *the-inv-into* (*carrier* *G*) *rep* *x*

and *h* = *the-inv-into* (*carrier* *G*) *rep* *y*

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

moreover from *g-def h-def* **have** *g* \in *carrier* *G* *h* \in *carrier* *G*

using *inj* *x y* *the-inv-into-into* **unfolding** *flatten-def* **by** (*metis partial-object.select-conv(1)*
subset-refl)+

hence $g \otimes_G h \in \text{carrier } G$ by (metis group group.is-monoid monoid.m-closed)
 hence $\text{rep } (g \otimes_G h) \in \text{carrier } (\text{flatten } G \text{ rep})$ 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 $\mathbf{1}_{\text{flatten } G \text{ rep}} \in \text{carrier } (\text{flatten } G \text{ rep})$ unfolding flatten-def by (simp
 add: group group.is-monoid)
 next
 fix $x \ y \ z$
 assume $x: x \in \text{carrier } (\text{flatten } G \text{ rep})$ and $y: y \in \text{carrier } (\text{flatten } G \text{ rep})$ and $z: z \in \text{carrier } (\text{flatten } G \text{ rep})$
 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 $x \otimes_{\text{flatten } G \text{ rep}} y \otimes_{\text{flatten } G \text{ rep}} z = (\text{rep } (g \otimes_G h)) \otimes_{\text{flatten } G \text{ rep}} z$
 unfolding flatten-def by auto
 also have $\dots = \text{rep } (\text{the-inv-into } (\text{carrier } G) \text{ rep } (\text{rep } (g \otimes_G h)) \otimes_G k)$ using
 $k\text{-def}$ unfolding flatten-def by auto
 also from $g\text{-def } h\text{-def } k\text{-def}$ 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-into unfolding flatten-def by fastforce+
 hence $gh: g \otimes_G h \in \text{carrier } G$ and $hk: h \otimes_G k \in \text{carrier } G$ by (metis group
 group.is-monoid monoid.m-closed)+
 hence $\text{rep } (\text{the-inv-into } (\text{carrier } G) \text{ rep } (\text{rep } (g \otimes_G h)) \otimes_G k) = \text{rep } ((g \otimes_G h)$
 $\otimes_G k)$
 unfolding flatten-def using inj the-inv-into-f-f by fastforce
 also have $\dots = \text{rep } (g \otimes_G (h \otimes_G k))$ using group group.is-monoid $ghkG$
 monoid.m-assoc by fastforce
 also have $\dots = x \otimes_{\text{flatten } G \text{ rep}} (\text{rep } (h \otimes_G k))$ unfolding $g\text{-def}$ flatten-def
 using hk inj the-inv-into-f-f by fastforce
 also have $\dots = x \otimes_{\text{flatten } G \text{ rep}} (y \otimes_{\text{flatten } G \text{ rep}} z)$ unfolding $h\text{-def } k\text{-def}$
 flatten-def using $x \ y$ by force
 finally show $x \otimes_{\text{flatten } G \text{ rep}} y \otimes_{\text{flatten } G \text{ rep}} z = x \otimes_{\text{flatten } G \text{ rep}} (y \otimes_{\text{flatten } G \text{ rep}} z)$.
 next
 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 $gG: g \in \text{carrier } G$ using inj x unfolding flatten-def using the-inv-into-into
 by force
 have $\mathbf{1}_G \in (\text{carrier } G)$ by (simp add: group group.is-monoid)
 hence $\text{the-inv-into } (\text{carrier } G) \text{ rep } (\mathbf{1}_{\text{flatten } G \text{ rep}}) = \mathbf{1}_G$ unfolding flatten-def
 using the-inv-into-f-f inj by force
 hence $\mathbf{1}_{\text{flatten } G \text{ rep}} \otimes_{\text{flatten } G \text{ rep}} x = \text{rep } (\mathbf{1}_G \otimes_G g)$ unfolding flatten-def
 $g\text{-def}$ by simp
 also have $\dots = \text{rep } g$ using gG group by (metis group.is-monoid monoid.l-one)
 also have $\dots = x$ unfolding $g\text{-def}$ using inj x f-the-inv-into-f unfolding flat-
 ten-def by force
 finally show $\mathbf{1}_{\text{flatten } G \text{ rep}} \otimes_{\text{flatten } G \text{ rep}} x = x$.

next
from *group inj* **have** $hom:rep \in hom\ G\ (flatten\ G\ rep)$ **using** *flatten-set-group-hom*
by *auto*
fix x
assume $x:x \in carrier\ (flatten\ G\ rep)$
define g **where** $g = the-inv-into\ (carrier\ G)\ rep\ x$
hence $gG:g \in carrier\ G$ **using** *inj x* **unfolding** *flatten-def* **using** *the-inv-into-into*
by *force*
hence $invG:inv_G\ g \in carrier\ G$ **by** (*metis group group.inv-closed*)
hence $rep\ (inv_G\ g) \in carrier\ (flatten\ G\ rep)$ **unfolding** *flatten-def* **by** *auto*
moreover **have** $rep\ (inv_G\ g) \otimes_{flatten\ G\ rep}\ x = rep\ (inv_G\ g) \otimes_{flatten\ G\ rep}\ (rep\ g)$
unfolding *g-def* **using** *f-the-inv-into-f inj x* **unfolding** *flatten-def* **by** *fastforce*
hence $rep\ (inv_G\ g) \otimes_{flatten\ G\ rep}\ x = rep\ (inv_G\ g \otimes_G\ g)$
using *hom* **unfolding** *hom-def* **using** *gG invG hom-def* **by** *auto*
hence $rep\ (inv_G\ g) \otimes_{flatten\ G\ rep}\ x = rep\ \mathbf{1}_G$ **using** *invG gG* **by** (*metis group group.l-inv*)
hence $rep\ (inv_G\ g) \otimes_{flatten\ G\ rep}\ x = \mathbf{1}_{flatten\ G\ rep}$ **unfolding** *flatten-def* **by** *auto*
ultimately show $\exists y \in carrier\ (flatten\ G\ rep). y \otimes_{flatten\ G\ rep}\ x = \mathbf{1}_{flatten\ G\ rep}$
by *auto*
qed

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

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

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

```

unfolding flatten-def apply (auto)
done

end

```

```

theory SimpleGroups
imports
  SubgroupsAndNormalSubgroups
  Secondary-Sylow.SndSylow
  SndIsomorphismGrp
begin

```

6 Simple Groups

```

locale simple-group = group +
  assumes order-gt-one: order  $G > 1$ 
  assumes no-real-normal-subgroup:  $\bigwedge H. H \triangleleft G \implies (H = \text{carrier } G \vee H = \{1\})$ 

```

```

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  $1 < \text{order } G$  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  $H = 1 \vee \text{card } H = \text{order } G$  unfolding prime-nat-iff by
simp
  thus  $H = \text{carrier } G \vee H = \{1\}$ 
  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  $H = \text{order } G$ 
    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:group H
  assumes iso:φ ∈ iso G H
  shows simple-group H
unfolding simple-group-def simple-group-axioms-def using assms(1)
proof (auto del: equalityI)
  from iso have order G = order H unfolding iso-def order-def using bij-betw-same-card
  by auto
  with order-gt-one show Suc 0 < order H by simp
next
  have inv-iso:(inv-into (carrier G) φ) ∈ iso H G using iso
  by (simp add: iso-set-sym)
  fix N
  assume NH:N ◁ H and Nneq1:N ≠ {1H}
  then interpret Nnormal: normal N H by simp
  define M where M = (inv-into (carrier G) φ) ‘ N
  hence MG:M ◁ G using inv-iso NH H by (metis is-group iso-normal-subgroup)
  have surj:φ ‘ carrier G = carrier H using iso unfolding iso-def bij-betw-def
  by simp
  hence MN:φ ‘ M = N unfolding M-def using Nnormal.subset image-inv-into-cancel
  by metis
  moreover have M ≠ {1}
  proof(rule notI)
    assume M = {1}
    hence φ ‘ M = {φ 1} by (metis (full-types) image-empty image-insert)
    hence φ ‘ M = {1H} by (metis (lifting) Nnormal.is-subgroup MN calculation
    singleton-iff subgroup.one-closed)
    thus False using Nneq1 MN by simp
  qed
  hence M = carrier G using no-real-normal-subgroup MG by auto
  ultimately show N = carrier H using surj by simp
qed

```

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

```

lemma (in group) self-factor-not-simple:¬ simple-group (G Mod (carrier G))
proof
  assume assm:simple-group (G Mod (carrier G))
  have group (G⟨carrier := {1}⟩) by (metis subgroup-imp-group triv-subgroup)
  with assm self-factor-iso simple-group.iso-simple have simple-group (G⟨carrier
:= {1}⟩) by auto
  thus False using simple-group.simple-not-triv by force
qed

```

end

```

theory MaximalNormalSubgroups
imports
  SubgroupsAndNormalSubgroups
  SimpleGroups
begin

```

7 Facts about maximal normal subgroups

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

```

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

```

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

```

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

```

proof

```

  assume max-normal-subgroup  $H \ G$ 
  then interpret maxH: 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)  $\neq \{1_{G \text{ Mod } H}\}$  using
    fact-group-trivial-iff by auto
    hence  $1 \neq \text{order } (G \text{ Mod } H)$  using factgroup-is-group group.order-one-triv-iff
  by metis
    with gt0 show  $1 < \text{order } (G \text{ Mod } H)$  unfolding order-def FactGroup-def by
    auto
  next
    fix  $A'$ 
    assume A'normal:  $A' \triangleleft G \text{ Mod } H$  and A'nottriv:  $A' \neq \{1_{G \text{ Mod } H}\}$ 
    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 un-
folding FactGroup-def by force
    hence  $H \subseteq A$  unfolding A-def by auto
    hence A1:  $H \triangleleft (G \setminus \text{carrier } := A)$  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 $A\text{-def}$ **using** $\text{factgroup-subgroup-union-factor } A'\text{normal normal-imp-subgroup}$
by auto
from $A1$ **interpret** $\text{normalHA: normal } H (G(\text{carrier} := A))$ **by** metis
have $H \subseteq A$ **using** $\text{normalHA.is-subgroup subgroup.subset}$ **by** force
with $A2$ **have** $A = H \vee A = \text{carrier } G$ **using** maxH.max-normal **by** auto
thus $A' = \text{carrier } (G \text{ Mod } H)$
proof
assume $A = H$
hence $\text{carrier } (G(\text{carrier} := A) \text{ Mod } H) = \{\mathbf{1}_{(G(\text{carrier} := A) \text{ Mod } H)}\}$
by $(\text{metis finite is-group normalHA.fact-group-trivial-iff normalHA.subgroup-self normalHA.subset subgroup-finite subgroup-of-restricted-group subgroup-of-subgroup subset-antisym})$
also **have** $\dots = \{\mathbf{1}_{G \text{ Mod } H}\}$ **unfolding** FactGroup-def **by** auto
finally **have** $A' = \{\mathbf{1}_{G \text{ Mod } H}\}$ **using** $A3$ **unfolding** FactGroup-def **by** simp
with $A'\text{nottriv}$ **show** $?thesis..$
next
assume $A = \text{carrier } G$
hence $(G(\text{carrier} := A) \text{ Mod } H) = G \text{ Mod } H$ **by** auto
thus $A' = \text{carrier } (G \text{ Mod } H)$ **using** $A3$ **unfolding** FactGroup-def **by** simp
qed
qed
next
assume $\text{simple:simple-group } (G \text{ Mod } H)$
show $\text{max-normal-subgroup } H G$
proof
from simple **have** $\text{carrier } (G \text{ Mod } H) \neq \{\mathbf{1}_{G \text{ Mod } H}\}$ **unfolding** $\text{simple-group-def simple-group-axioms-def order-def}$ **by** auto
with $\text{finite fact-group-trivial-iff}$ **show** $H \neq \text{carrier } G$ **by** auto
next
fix A
assume $A:A \triangleleft G \ A \neq H \ A \neq \text{carrier } G$
show $\neg H \subseteq A$
proof
assume $HA:H \subseteq A$
hence $H \triangleleft (G(\text{carrier} := A))$ **by** $(\text{metis } A(1) \text{ inv-op-closed2 is-subgroup normal-inv-iff normal-restrict-supergroup})$
then **interpret** $\text{normalHA: normal } H (G(\text{carrier} := A))$ **by** simp
from finite **have** $\text{finiteA:finite } A$ **using** $A(1) \text{ normal-imp-subgroup}$ **by** $(\text{metis subgroup-finite})$
have $\text{rcosets}_{(G(\text{carrier} := A))} H \triangleleft G \text{ Mod } H$ **using** $\text{normality-factorization is-normal HA } A(1)$ **by** auto
with simple **have** $\text{rcosets}_{(G(\text{carrier} := A))} H = \{\mathbf{1}_{G \text{ Mod } H}\} \vee \text{rcosets}_{(G(\text{carrier} := A))} H = \text{carrier } (G \text{ Mod } H)$
unfolding $\text{simple-group-def simple-group-axioms-def}$ **by** auto
thus False
proof
assume $\text{rcosets}_{G(\text{carrier} := A)} H = \{\mathbf{1}_{G \text{ Mod } H}\}$

```

    hence rcosets $G(\text{carrier} := A)$   $H = \{1(G(\text{carrier} := A)) \text{ Mod } H\}$  unfolding
    FactGroup-def by auto
    with finiteA have  $H = A$  using normalHA.fact-group-trivial-iff unfolding
    FactGroup-def by auto
    with A(2) show ?thesis by simp
  next
    assume AHGH:rcosets $G(\text{carrier} := A)$   $H = \text{carrier } (G \text{ Mod } H)$ 
    have  $A = \text{carrier } G$  unfolding FactGroup-def RCOSETS-def
    proof
      show  $A \subseteq \text{carrier } G$  using A(1) normal-imp-subgroup subgroup.subset by
      metis
    next
      show  $\text{carrier } G \subseteq A$ 
      proof
        fix x
        assume x:x ∈ carrier G
        hence  $H \#> x \in \text{rcosets } H$  unfolding RCOSETS-def by auto
        with AHGH have  $H \#> x \in \text{rcosets}_{G(\text{carrier} := A)} H$  unfolding
        FactGroup-def by simp
        then obtain x' where x':x' ∈ A  $H \#> x = H \#>_{G(\text{carrier} := A)} x'$ 
unfolding RCOSETS-def by auto
        hence  $H \#> x = H \#> x'$  unfolding r-coset-def by auto
        hence  $x \in H \#> x'$  by (metis is-subgroup rcos-self x)
        hence  $x \in A \#> x'$  using HA 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 *fin:finite* (carrier G)
assumes $\exists! Q. Q \in \text{subgroups-of-size } q$
shows (*THE* $Q. Q \in \text{subgroups-of-size } q$) $\triangleleft G$
proof –
from *assms* **obtain** Q **where** $Q \in \text{subgroups-of-size } q$ **by** *auto*
define Q **where** $Q = (\text{THE } Q. Q \in \text{subgroups-of-size } q)$
with *assms* **have** $Q\text{size}: Q \in \text{subgroups-of-size } q$ **using** *theI* **by** *metis*
hence $QG:\text{subgroup } Q \ G$ **and** $\text{card}Q:\text{card } Q = q$ **unfolding** *subgroups-of-size-def*
by *auto*
from QG **have** $Q \triangleleft G$ **apply**(*rule normalI*)
proof
fix g
assume $g:g \in \text{carrier } G$
hence $\text{inv}g:\text{inv } g \in \text{carrier } G$ **by** (*metis inv-closed*)
with *fin* $Q\text{size}$ **have** *conjugation-action* q (*inv* g) $Q \in \text{subgroups-of-size } q$ **by**
(*metis conjugation-is-size-invariant*)
with g $Q\text{size}$ **have** (*inv* g) $<\# (Q \#> \text{inv } (\text{inv } g)) \in \text{subgroups-of-size } q$
unfolding *conjugation-action-def* **by** *auto*
with $\text{inv}g$ g **have** $\text{inv } g <\# (Q \#> g) = Q$ **by** (*metis Qsize assms(2) inv-inv*)
with QG QG g **show** $Q \#> g = g <\# Q$ **by** (*rule conj-wo-inv*)
qed
with $Q\text{-def}$ **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 $\exists! Q. Q \in (\text{subgroups-of-size } q)$
proof –
from *primep primeq pq* **have** $nq\text{dvd}p:\neg (q \text{ dvd } p)$ **by** (*metis less-not-refl3 prime-nat-iff*)
define *calM* **where** $\text{cal}M = \{s. s \subseteq \text{carrier } G \wedge \text{card } s = q \wedge 1\}$
define *RelM* **where** $\text{Rel}M = \{(N1, N2). N1 \in \text{cal}M \wedge N2 \in \text{cal}M \wedge (\exists g \in \text{carrier } G. N1 = N2 \#> g)\}$
interpret *syl*: *snd-sylow* G q 1 p *calM* *RelM*
unfolding *snd-sylow-def sylow-def snd-sylow-axioms-def sylow-axioms-def*
using *is-group primeq orderG finite nq\text{dvd}p calM-def RelM-def* **by** *auto*
obtain Q **where** $Q:Q \in \text{subgroups-of-size } q$ **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 $P:P \in \text{subgroups-of-size } q$
have $\text{card } (\text{subgroups-of-size } q) \bmod q = 1$ **by** (*metis power-one-right syl.p-sylow-mod-p*)

moreover **have** $\text{card } (\text{subgroups-of-size } q) \text{ dvd } p$ **by** (*metis power-one-right*

syl.num-sylow-dvd-remainder)
then have $\text{card}(\text{subgroups-of-size } q) = p \vee \text{card}(\text{subgroups-of-size } q) = 1$
using *primep* **by** (*auto simp add: prime-nat-iff*)
ultimately have $\text{card}(\text{subgroups-of-size } q) = 1$ **using** *pq*
by *auto*
with $Q P$ **show** $P = Q$ **by** (*auto simp: card-Suc-eq*)
qed
qed

... And this unique subgroup is normal.

corollary (*in group*) *pq-order-subgrp-normal*:
assumes *finite:finite* (*carrier G*)
assumes *orderG:order* $G = q * p$
assumes *primep:prime* p **and** *primeq:prime* q **and** *pq:p < q*
shows (*THE Q. Q ∈ subgroups-of-size q*) $\triangleleft G$
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 \mathfrak{G} 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 \mathfrak{G}
assumes *notempty:* $\mathfrak{G} \neq []$
assumes *hd:hd* $\mathfrak{G} = \{1\}$
assumes *last:last* $\mathfrak{G} = \text{carrier } G$
assumes *normal:* $\bigwedge i. i + 1 < \text{length } \mathfrak{G} \implies (\mathfrak{G} ! i) \triangleleft G(\text{carrier} := \mathfrak{G} ! (i + 1))$

lemma (*in normal-series*) *is-normal-series: normal-series G* \mathfrak{G} **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  $\mathfrak{G} = 2$ 
  shows  $\mathfrak{G} = [\{1\}, \text{carrier } G]$ 
proof(rule nth-equalityI)
  from assms show length  $\mathfrak{G} = \text{length } [\{1\}, \text{carrier } G]$  by auto
next
  show  $\mathfrak{G} ! i = [\{1\}, \text{carrier } G] ! i$  if  $i < \text{length } \mathfrak{G}$  for  $i$ 
  proof -
    have  $i = 0 \vee i = 1$  using that assms by auto
    thus  $\mathfrak{G} ! i = [\{1\}, \text{carrier } G] ! i$ 
    proof(rule disjE)
      assume  $i : i = 0$ 
      hence  $\mathfrak{G} ! i = \text{hd } \mathfrak{G}$  by (metis hd-conv-nth notempty)
      thus  $\mathfrak{G} ! i = [\{1\}, \text{carrier } G] ! i$  using hd i by simp
    next
      assume  $i : i = 1$ 
      with assms have  $\mathfrak{G} ! i = \text{last } \mathfrak{G}$  by (metis diff-add-inverse last-conv-nth
nat-1-add-1 notempty)
      thus  $\mathfrak{G} ! i = [\{1\}, \text{carrier } G] ! i$  using last i by simp
    qed
  qed
qed

```

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

```

lemma (in group) normal-series-extend:
  assumes normal:normal-series ( $G \langle \text{carrier} := H \rangle$ )  $\mathfrak{H}$ 
  assumes  $HG : H \triangleleft G$ 
  shows normal-series  $G$  ( $\mathfrak{H} @ [\text{carrier } G]$ )
proof -
  from normal interpret normalH: normal-series ( $G \langle \text{carrier} := H \rangle$ )  $\mathfrak{H}$ .
  from normalH.hd have hd  $\mathfrak{H} = \{1\}$  by simp
  with normalH.notempty have hdTriv:hd ( $\mathfrak{H} @ [\text{carrier } G]$ ) =  $\{1\}$  by (metis
hd-append2)
  show ?thesis unfolding normal-series-def normal-series-axioms-def using is-group
  proof auto
    fix  $x$ 
    assume  $x \in \text{hd } (\mathfrak{H} @ [\text{carrier } G])$ 
    with hdTriv show  $x = 1$  by simp
  next
    from hdTriv show  $1 \in \text{hd } (\mathfrak{H} @ [\text{carrier } G])$  by simp
  next
    fix  $i$ 
    assume  $i : i < \text{length } \mathfrak{H}$ 
    show ( $\mathfrak{H} @ [\text{carrier } G]$ ) !  $i \triangleleft G \langle \text{carrier} := (\mathfrak{H} @ [\text{carrier } G]) ! \text{Suc } i \rangle$ 
    proof (cases  $i + 1 < \text{length } \mathfrak{H}$ )
      case True
      with normalH.normal have  $\mathfrak{H} ! i \triangleleft G \langle \text{carrier} := \mathfrak{H} ! (i + 1) \rangle$  by auto

```

with i **have** $(\mathfrak{H} @ [carrier\ G]) ! i \triangleleft G(\text{carrier} := \mathfrak{H} ! (i + 1))$ **using**
nth-append by metis
with *True* **show** $(\mathfrak{H} @ [carrier\ G]) ! i \triangleleft G(\text{carrier} := (\mathfrak{H} @ [carrier\ G]) !$
 $(Suc\ i))$ **using** *nth-append Suc-eq-plus1 by metis*
next
case *False*
with i **have** $i2:i + 1 = length\ \mathfrak{H}$ **by** *simp*
from i **have** $(\mathfrak{H} @ [carrier\ G]) ! i = \mathfrak{H} ! i$ **by** *(metis nth-append)*
also from $i2$ *normalH.notempty* **have** $\dots = last\ \mathfrak{H}$ **by** *(metis add-diff-cancel-right'*
last-conv-nth)
also from *normalH.last* **have** $\dots = H$ **by** *simp*
finally have $(\mathfrak{H} @ [carrier\ G]) ! i = H$.
moreover from $i2$ **have** $(\mathfrak{H} @ [carrier\ G]) ! (i + 1) = carrier\ G$ **by** *(metis*
nth-append-length)
ultimately show *?thesis* **using** *HG by auto*
qed
qed
qed

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

lemma (in *normal-series*) *normal-series-subgroups*:

shows $i < length\ \mathfrak{G} \implies subgroup\ (\mathfrak{G} ! i)\ G$

proof –

have $i + 1 < length\ \mathfrak{G} \implies subgroup\ (\mathfrak{G} ! i)\ G$

proof (*induction length\ \mathfrak{G} - (i + 2) arbitrary: i*)

case 0

hence $i:i + 2 = length\ \mathfrak{G}$ **by** *simp*

hence $ii:i + 1 = length\ \mathfrak{G} - 1$ **by** *force*

from i *normal* **have** $\mathfrak{G} ! i \triangleleft G(\text{carrier} := \mathfrak{G} ! (i + 1))$ **by** *auto*

with ii *last notempty* **show** $subgroup\ (\mathfrak{G} ! i)\ G$ **using** *last-conv-nth normal-imp-subgroup by fastforce*

next

case (*Suc k*)

from *Suc(3)* *normal* **have** $i:subgroup\ (\mathfrak{G} ! i)\ (G(\text{carrier} := \mathfrak{G} ! (i + 1)))$

using *normal-imp-subgroup by auto*

from *Suc(2)* **have** $k:k = length\ \mathfrak{G} - ((i + 1) + 2)$ **by** *arith*

with *Suc* **have** $subgroup\ (\mathfrak{G} ! (i + 1))\ G$ **by** *simp*

with i **show** $subgroup\ (\mathfrak{G} ! i)\ G$ **by** *(metis is-group subgroup.subgroup-of-subgroup)*

qed

moreover have $i + 1 = length\ \mathfrak{G} \implies subgroup\ (\mathfrak{G} ! i)\ G$

using *last notempty last-conv-nth by (metis add-diff-cancel-right' subgroup-self)*

ultimately show $i < length\ \mathfrak{G} \implies subgroup\ (\mathfrak{G} ! i)\ G$ **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 $\mathfrak{G} ! (length\ \mathfrak{G} - 2) \triangleleft G$

proof (*cases 2 ≤ length\ \mathfrak{G}*)

case *False*

```

with notempty have length:length  $\mathfrak{G} = 1$  by (metis Suc-eq-plus1 leI length-0-conv
less-2-cases plus-nat.add-0)
with hd have  $\mathfrak{G} ! (\text{length } \mathfrak{G} - 2) = \{1\}$  using hd-conv-nth notempty by auto
with length show ?thesis by (metis trivial-subgroup-is-normal)
next
  case True
  hence  $(\text{length } \mathfrak{G} - 2) + 1 < \text{length } \mathfrak{G}$  by arith
  with normal last have  $\mathfrak{G} ! (\text{length } \mathfrak{G} - 2) \triangleleft G(\text{carrier} := \mathfrak{G} ! ((\text{length } \mathfrak{G} - 2) + 1))$  by auto
  have  $1 + (1 + (\text{length } \mathfrak{G} - (1 + 1))) = \text{length } \mathfrak{G}$ 
  using True le-add-diff-inverse by presburger
  then have  $\mathfrak{G} ! (\text{length } \mathfrak{G} - 2) \triangleleft G(\text{carrier} := \mathfrak{G} ! (\text{length } \mathfrak{G} - 1))$ 
  by (metis  $\langle \mathfrak{G} ! (\text{length } \mathfrak{G} - 2) \triangleleft G(\text{carrier} := \mathfrak{G} ! (\text{length } \mathfrak{G} - 2 + 1)) \rangle$ 
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  $i \leq \text{length } \mathfrak{G}$  and  $0 < i$ 
  shows normal-series  $(G(\text{carrier} := \mathfrak{G} ! (i - 1)))$  (take i  $\mathfrak{G}$ )
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  $pq:p < q$ 
  shows normal-series  $G [\{1\}, (\text{THE } H. H \in \text{subgroups-of-size } q), \text{carrier } G]$ 
proof -
  define  $H$  where  $H = (\text{THE } H. H \in \text{subgroups-of-size } q)$ 
  with assms have  $HG:H \triangleleft G$  by (metis pq-order-subgrp-normal)
  then interpret groupH: group  $G(\text{carrier} := H)$  unfolding normal-def by
(metis subgroup-imp-group)
  have normal-series  $(G(\text{carrier} := H)) [\{1\}, H]$  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 $quotients = \text{map } (\lambda i. G(\text{carrier} := \mathfrak{G}! (i + 1))) \text{ Mod } \mathfrak{G}! i [0..<((\text{length } \mathfrak{G}) - 1)]$

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

lemma (in *normal-series*) *quotients-length*:
 shows $\text{length } quotients + 1 = \text{length } \mathfrak{G}$
proof –
 have $\text{length } quotients + 1 = \text{length } [0..<((\text{length } \mathfrak{G}) - 1)] + 1$ **unfolding**
quotients-def **by** *simp*
 also have $\dots = (\text{length } \mathfrak{G} - 1) + 1$ **by** (*metis diff-zero length-upt*)
 also with *notempty* have $\dots = \text{length } \mathfrak{G}$
 by (*simp add: ac-simps*)
 finally show *?thesis* .
qed

lemma (in *normal-series*) *last-quotient*:
 assumes $\text{length } \mathfrak{G} > 1$
 shows $\text{last } quotients = G \text{ Mod } \mathfrak{G}! (\text{length } \mathfrak{G} - 1 - 1)$
proof –
 from *assms* have $l\text{simp}:\text{length } \mathfrak{G} - 1 - 1 + 1 = \text{length } \mathfrak{G} - 1$ **by** *auto*
 from *assms* have $quotients \neq []$ **unfolding** *quotients-def* **by** *auto*
 hence $\text{last } quotients = quotients! (\text{length } quotients - 1)$ **by** (*metis last-conv-nth*)
 also have $\dots = quotients! (\text{length } \mathfrak{G} - 1 - 1)$ **by** (*metis add-diff-cancel-left'*
quotients-length add commute)
 also have $\dots = G(\text{carrier} := \mathfrak{G}! ((\text{length } \mathfrak{G} - 1 - 1) + 1)) \text{ Mod } \mathfrak{G}! (\text{length } \mathfrak{G} - 1 - 1)$
unfolding *quotients-def* **using** *assms* **by** *auto*
 also have $\dots = G(\text{carrier} := \mathfrak{G}! (\text{length } \mathfrak{G} - 1)) \text{ Mod } \mathfrak{G}! (\text{length } \mathfrak{G} - 1 - 1)$ **using** *lsimp* **by** *simp*
 also have $\dots = G \text{ Mod } \mathfrak{G}! (\text{length } \mathfrak{G} - 1 - 1)$ **using** *last last-conv-nth notempty*
by *force*
 finally show *?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 *normal-series* $H (\text{map } (\text{image } \Psi) \mathfrak{G})$
apply (*simp add: normal-series-def normal-series-axioms-def*)
using H *notempty* **apply** *simp*
proof (*rule conjI*)
 from H *is-group iso* have *group-hom:group-hom* $G H \Psi$ **unfolding** *group-hom-def*
group-hom-axioms-def iso-def **by** *auto*
 have $\text{hd } (\text{map } (\text{image } \Psi) \mathfrak{G}) = \Psi \text{ ' } \{1\}$ **by** (*metis hd-map hd notempty*)
 also have $\dots = \{\Psi 1\}$ **by** (*metis image-empty image-insert*)
 also have $\dots = \{1_H\}$ **using** *group-hom group-hom.hom-one* **by** *auto*

finally show $hd (map ((\cdot) \Psi) \mathfrak{G}) = \{1_H\}$.
next
show $last (map ((\cdot) \Psi) \mathfrak{G}) = carrier H \wedge (\forall i. Suc i < length \mathfrak{G} \longrightarrow \Psi ' \mathfrak{G} ! i \triangleleft H \langle carrier := \Psi ' \mathfrak{G} ! Suc i \rangle)$
proof (*auto del: equalityI*)
have $last (map ((\cdot) \Psi) \mathfrak{G}) = \Psi ' (carrier G)$ **using** *last last-map notempty by metis*
also have $\dots = carrier H$ **using** *iso unfolding iso-def bij-betw-def by simp*
finally show $last (map ((\cdot) \Psi) \mathfrak{G}) = carrier H$.
next
fix i
assume $i: Suc i < length \mathfrak{G}$
hence $norm: \mathfrak{G} ! i \triangleleft G \langle carrier := \mathfrak{G} ! Suc i \rangle$ **using** *normal by simp*
moreover have $restrict \Psi (\mathfrak{G} ! Suc i) \in iso (G \langle carrier := \mathfrak{G} ! Suc i \rangle)$
($H \langle carrier := \Psi ' \mathfrak{G} ! Suc i \rangle$)
by (*metis H i is-group iso iso-restrict normal-series-subgroups*)
moreover have $group (G \langle carrier := \mathfrak{G} ! Suc i \rangle)$ **by** (*metis i normal-series-subgroups subgroup-imp-group*)
moreover hence $subgroup (\mathfrak{G} ! Suc i) G$ **by** (*metis i normal-series-subgroups*)
hence $subgroup (\Psi ' \mathfrak{G} ! Suc i) H$ **by** (*metis H is-group iso iso-subgroup*)
hence $group (H \langle carrier := \Psi ' \mathfrak{G} ! Suc i \rangle)$ **by** (*metis H subgroup.subgroup-is-group*)
ultimately have $restrict \Psi (\mathfrak{G} ! Suc i) ' \mathfrak{G} ! i \triangleleft H \langle carrier := \Psi ' \mathfrak{G} ! Suc i \rangle$
using *is-group H iso-normal-subgroup by (auto cong del: image-cong-simp)*
moreover from *norm* **have** $\mathfrak{G} ! i \subseteq \mathfrak{G} ! Suc i$ **unfolding** *normal-def subgroup-def* **by** *auto*
hence $\{y. \exists x \in \mathfrak{G} ! i. y = (if x \in \mathfrak{G} ! Suc i then \Psi x else undefined)\} = \{y. \exists x \in \mathfrak{G} ! i. y = \Psi x\}$ **by** *auto*
ultimately show $\Psi ' \mathfrak{G} ! i \triangleleft H \langle carrier := \Psi ' \mathfrak{G} ! Suc i \rangle$ **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: $\wedge i. i + 1 < length \mathfrak{G} \implies simple-group (G \langle carrier := \mathfrak{G} ! (i + 1) \rangle Mod \mathfrak{G} ! i)$*

lemma (*in composition-series*) *is-composition-series:*
shows *composition-series G \mathfrak{G}*
by (*rule composition-series-axioms*)

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

lemma (*in composition-series*) *composition-series-length-one:*
shows $(length \mathfrak{G} = 1) = (\mathfrak{G} = \{1\})$

proof
 assume $\text{length } \mathfrak{G} = 1$
 with hd have $\text{length } \mathfrak{G} = \text{length } [\{1\}] \wedge (\forall i < \text{length } \mathfrak{G}. \mathfrak{G} ! i = [\{1\}] ! i)$ using
 $hd\text{-conv-nth notempty}$ by $force$
 thus $\mathfrak{G} = [\{1\}]$ using $list\text{-eq-iff-nth-eq}$ by $blast$
next
 assume $\mathfrak{G} = [\{1\}]$
 thus $\text{length } \mathfrak{G} = 1$ by $simp$
qed

lemma (in *composition-series*) *composition-series-triv-group*:

shows $(\text{carrier } G = \{1\}) = (\mathfrak{G} = [\{1\}])$

proof

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

have $\text{length } \mathfrak{G} = 1$

proof (rule *ccontr*)

assume $\text{length } \mathfrak{G} \neq 1$

with $notempty$ have $\text{length}:\text{length } \mathfrak{G} \geq 2$ by (*metis Suc-eq-plus1 length-0-conv less-2-cases not-less plus-nat.add-0*)

with $simplefact\ hd\ hd\text{-conv-nth notempty}$ have $simple\text{-group } (G(\text{carrier} := \mathfrak{G} ! 1) \text{ Mod } \{1\})$ by $force$

moreover have $SG:\text{subgroup } (\mathfrak{G} ! 1) G$ using $length\ normal\text{-series-subgroups}$
 by $auto$

hence $group (G(\text{carrier} := \mathfrak{G} ! 1))$ by (*metis subgroup-imp-group*)

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

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

ultimately show $False$ using $simple\text{-group.simple-not-triv}$ by $force$

qed

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

next

assume $\mathfrak{G} = [\{1\}]$

with $last$ show $\text{carrier } G = \{1\}$ 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 $i + 1 < \text{length } \mathfrak{G}$

assumes $i > 0$

shows $\mathfrak{G} ! i \neq \{1\}$

proof

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

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

using $simplefact$ by $auto$

assume $i:\mathfrak{G} ! i = \{1\}$

moreover from $assms$ have $(i - 1) + 1 = i$ by $auto$

ultimately have $G(\text{carrier} := \mathfrak{G} ! ((i - 1) + 1)) \text{ Mod } \mathfrak{G} ! (i - 1) = G(\text{carrier}$

```

:= {1}) Mod  $\mathfrak{G} ! (i - 1)$  using  $i$  by auto
  hence order (G⟦carrier :=  $\mathfrak{G} ! ((i - 1) + 1)$ ⟧ Mod  $\mathfrak{G} ! (i - 1)$ ) = 1 unfolding
FactGroup-def order-def RCOSETS-def by force
  thus False using  $i$  simple unfolding simple-group-def simple-group-axioms-def
by auto
qed

```

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

lemma (in *composition-series*) *composition-series-simple-group*:

shows (simple-group G) = ($\mathfrak{G} = [\{1\}$, carrier G])

proof

assume $\mathfrak{G} = [\{1\}$, carrier G]

with *simplefact* have simple-group (G Mod $\{1\}$) by auto

moreover have the-elem \in 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:simple-group* G

have length $\mathfrak{G} > 1$

proof (rule *ccontr*)

assume $\neg 1 < \text{length } \mathfrak{G}$

hence length $\mathfrak{G} = 1$ by (*simp add: Suc-leI antisym notempty*)

hence carrier $G = \{1\}$ using *hd last* by (metis *composition-series-length-one composition-series-triv-group*)

hence order $G = 1$ unfolding *order-def* by auto

with *simple* show False unfolding *simple-group-def simple-group-axioms-def* by auto

qed

moreover have length $\mathfrak{G} \leq 2$

proof (rule *ccontr*)

define k where $k = \text{length } \mathfrak{G} - 2$

assume $\neg (\text{length } \mathfrak{G} \leq 2)$

hence *gt2:length* $\mathfrak{G} > 2$ by *simp*

hence *ksmall:k + 1 < length* \mathfrak{G} unfolding *k-def* by auto

from *gt2* have *carrier:G ! (k + 1) = carrier* G using *notempty last last-conv-nth k-def*

by (metis *Nat.add-diff-assoc Nat.diff-cancel* $\langle \neg \text{length } \mathfrak{G} \leq 2 \rangle$ *add commute nat-le-linear one-add-one*)

from *normal ksmall* have $\mathfrak{G} ! k \triangleleft G$ ⟦carrier := $\mathfrak{G} ! (k + 1)$ ⟧ by *simp*

from *simplefact ksmall* have *simplek:simple-group* (G ⟦carrier := $\mathfrak{G} ! (k + 1)$ ⟧ Mod $\mathfrak{G} ! k$) by *simp*

from *simplefact ksmall* have *simplek':simple-group* (G ⟦carrier := $\mathfrak{G} ! ((k - 1) + 1)$ ⟧ Mod $\mathfrak{G} ! (k - 1)$) by auto

have $\mathfrak{G} ! k \triangleleft G$ using *carrier k-def gt2 normal ksmall* by force

with *simple* have $(\mathfrak{G} ! k) = \text{carrier } G \vee (\mathfrak{G} ! k) = \{1\}$ unfolding *simple-group-def simple-group-axioms-def* by *simp*

thus False

proof (rule *disjE*)

assume $\mathfrak{G} ! k = \text{carrier } G$

hence G ⟦carrier := $\mathfrak{G} ! (k + 1)$ ⟧ Mod $\mathfrak{G} ! k = G$ Mod (carrier G) using

```

carrier by auto
  with simplek self-factor-not-simple show False by auto
next
  assume  $\mathfrak{G} ! k = \{1\}$ 
  with ksmall k-def gt2 show False using inner-elements-not-triv by auto
qed
qed
ultimately have length  $\mathfrak{G} = 2$  by simp
thus  $\mathfrak{G} = [\{1\}, \text{carrier } G]$  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  $\mathfrak{G}$ 
shows  $\mathfrak{G} ! i \neq \mathfrak{G} ! (i + 1)$ 
proof
  from finite have finite ( $\mathfrak{G} ! (i + 1)$ )
  using i normal-series-subgroups subgroup.subset rev-finite-subset by metis
  hence fin:finite (carrier ( $G \lfloor \text{carrier} := \mathfrak{G} ! (i + 1) \rfloor$ )) by auto
  from i have norm: $\mathfrak{G} ! i \triangleleft (G \lfloor \text{carrier} := \mathfrak{G} ! (i + 1) \rfloor)$  by (rule normal)
  assume  $\mathfrak{G} ! i = \mathfrak{G} ! (i + 1)$ 
  hence  $\mathfrak{G} ! i = \text{carrier} (G \lfloor \text{carrier} := \mathfrak{G} ! (i + 1) \rfloor)$  by auto
  hence carrier (( $G \lfloor \text{carrier} := (\mathfrak{G} ! (i + 1)) \rfloor$ ) Mod ( $\mathfrak{G} ! i$ )) =  $\{1_{(G \lfloor \text{carrier} := \mathfrak{G} ! (i + 1) \rfloor) \text{ Mod } \mathfrak{G} ! i}\}$ 
  using norm fin normal.fact-group-trivial-iff by metis
  hence  $\neg$  simple-group (( $G \lfloor \text{carrier} := (\mathfrak{G} ! (i + 1)) \rfloor$ ) Mod ( $\mathfrak{G} ! i$ )) by (metis
simple-group.simple-not-triv)
  thus False by (metis i simplefact)
qed

```

The normal series for groups of order $p * q$ 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  $G [\{1\}, (\text{THE } H. H \in \text{subgroups-of-size } q), \text{carrier } G]$ 
unfolding composition-series-def composition-series-axioms-def
apply(auto)
using assms apply(rule pq-order-normal-series)
proof -
  define  $H$  where  $H = (\text{THE } H. H \in \text{subgroups-of-size } q)$ 
  from assms have exi: $\exists ! Q. Q \in (\text{subgroups-of-size } q)$  by (auto simp: pq-order-unique-subgrp)
  hence Hsize: $H \in \text{subgroups-of-size } q$  unfolding H-def using theI' by metis
  hence HsubG:subgroup  $H G$  unfolding subgroups-of-size-def by auto
  then interpret Hgroup: group  $G \lfloor \text{carrier} := H \rfloor$  by (metis subgroup-imp-group)
  fix  $i$ 
  assume  $i < \text{Suc } (\text{Suc } 0)$ 
  hence  $i = 0 \vee i = 1$  by auto
  thus simple-group ( $G \lfloor \text{carrier} := [H, \text{carrier } G] ! i \rfloor$  Mod  $[\{1\}, H, \text{carrier } G] ! i$ )

```

```

proof
  assume  $i:i = 0$ 
  from  $Hsize$  have  $orderH:order (G \setminus carrier := H) = q$  unfolding subgroups-of-size-def
order-def by simp
  hence  $order (G \setminus carrier := H) Mod \{1\} = q$  unfolding FactGroup-def using
card-rcosets-triv order-def
  by (metis Hgroup.card-rcosets-triv HsubG finite monoid.cases-scheme monoid.select-convs(2))
partial-object.select-convs(1) partial-object.update-convs(1) subgroup-finite)
  have  $normal \{1\} (G \setminus carrier := H)$  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  $group (G \setminus carrier := H) Mod \{1\}$  by (metis normal.factorgroup-is-group)
  with  $orderH$   $primeq$  have  $simple-group (G \setminus carrier := H) Mod \{1\}$  by (metis
 $\langle order (G \setminus carrier := H) Mod \{1\} = q \rangle$  group.prime-order-simple)
  with  $i$  show  $?thesis$  by simp
next
  assume  $i:i = 1$ 
  from  $assms$   $exi$  have  $H \triangleleft G$  unfolding  $H-def$  by (metis pq-order-subgrp-normal)
  hence  $groupGH:group (G Mod H)$  by (metis normal.factorgroup-is-group)
  from  $primeq$  have  $q \neq 0$  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  $\langle q \neq 0 \rangle$  have  $card (rcosets H) = p$  by auto
  hence  $order (G Mod H) = p$  unfolding order-def FactGroup-def by auto
  with  $groupGH$   $primep$  have  $simple-group (G Mod H)$  by (metis group.prime-order-simple)
  with  $i$  show  $?thesis$  by auto
qed
qed

```

Prefixes of composition series are also composition series.

lemma (*in composition-series*) *composition-series-prefix-closed*:

assumes $i \leq length \mathfrak{G}$ **and** $0 < i$

shows *composition-series* $(G \setminus carrier := \mathfrak{G} ! (i - 1))$ (*take* $i \mathfrak{G}$)

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

proof *auto*

from $assms$ **show** *normal-series* $(G \setminus carrier := \mathfrak{G} ! (i - Suc 0))$ (*take* $i \mathfrak{G}$) **by**
(*metis One-nat-def normal-series-prefix-closed*)

next

fix j

assume $j:Suc j < length \mathfrak{G}$ $Suc j < i$

with *simplefact* **show** *simple-group* $(G \setminus carrier := \mathfrak{G} ! Suc j) Mod \mathfrak{G} ! j$ **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 $2 \leq length \mathfrak{G}$

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

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

lemma (in *composition-series*) *composition-snd-simple-iff*:

assumes $i < \text{length } \mathfrak{G}$
shows (*simple-group* ($G(\text{carrier} := \mathfrak{G} ! i)$)) = ($i = 1$)
proof
assume *simpI: simple-group* ($G(\text{carrier} := \mathfrak{G} ! i)$)
hence $\mathfrak{G} ! i \neq \{1\}$ **using** *simple-group.simple-not-triv* **by** *force*
hence $i \neq 0$ **using** *hd hd-conv-nth notempty* **by** *auto*
then interpret *compTake: composition-series* $G(\text{carrier} := \mathfrak{G} ! i)$ *take* (*Suc* i) \mathfrak{G}
using *assms composition-series-prefix-closed* **by** (*metis diff-Suc-1 less-eq-Suc-le zero-less-Suc*)
from *simpI* **have** (*take* (*Suc* i) \mathfrak{G}) = [$\{1_{G(\text{carrier} := \mathfrak{G} ! i)}\}$, *carrier* ($G(\text{carrier} := \mathfrak{G} ! i)$)]
by (*metis compTake.composition-series-simple-group*)
hence *length* (*take* (*Suc* i) \mathfrak{G}) = 2 **by** *auto*
hence $\min(\text{length } \mathfrak{G}) (\text{Suc } i) = 2$ **by** (*metis length-take*)
with *assms* **have** $\text{Suc } i = 2$ **by** *force*
thus $i = 1$ **by** *simp*
next
assume $i : i = 1$
with *assms* **have** $2 \leq \text{length } \mathfrak{G}$ **by** *simp*
with i **show** *simple-group* ($G(\text{carrier} := \mathfrak{G} ! i)$) **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:finite* (*carrier* G)
assumes *length:length* $\mathfrak{G} > 1$
shows *max-normal-subgroup* ($\mathfrak{G} ! (\text{length } \mathfrak{G} - 2)$) G
unfolding *max-normal-subgroup-def max-normal-subgroup-axioms-def*
proof (*auto del: equalityI*)
show $\mathfrak{G} ! (\text{length } \mathfrak{G} - 2) \triangleleft G$ **by** (*rule normal-series-snd-to-last*)
next

```

define  $G'$  where  $G' = \mathfrak{G} ! (\text{length } \mathfrak{G} - 2)$ 
from  $\text{length}$  have  $\text{length21}:\text{length } \mathfrak{G} - 2 + 1 = \text{length } \mathfrak{G} - 1$  by arith
from  $\text{length}$  have  $\text{length } \mathfrak{G} - 2 + 1 < \text{length } \mathfrak{G}$  by arith
with simplefact have simple-group ( $G \setminus \text{carrier} := \mathfrak{G} ! ((\text{length } \mathfrak{G} - 2) + 1)$ )
Mod  $G'$ ) unfolding  $G'$ -def by auto
with  $\text{length21}$  have simple-last:simple-group ( $G \text{ Mod } G'$ ) using last notempty
last-conv-nth by fastforce
{
  assume snd-to-last-eq: $G' = \text{carrier } G$ 
  hence  $\text{carrier } (G \text{ Mod } G') = \{\mathbf{1}_{G \text{ Mod } G'}\}$ 
  using normal-series-snd-to-last finite normal.fact-group-trivial-iff unfolding
 $G'$ -def by metis
  with snd-to-last-eq have  $\neg \text{simple-group } (G \text{ Mod } G')$  by (metis self-factor-not-simple)
  with simple-last show False unfolding  $G'$ -def by auto
}
{
  have  $G'G:G' \triangleleft G$  unfolding  $G'$ -def by (rule normal-series-snd-to-last)
  fix  $J$ 
  assume  $J:J \triangleleft G \ J \neq G' \ J \neq \text{carrier } G \ G' \subseteq J$ 
  hence  $JG'GG':\text{rcosets}_{(G \setminus \text{carrier} := J)} G' \triangleleft G \text{ Mod } G'$  using normality-factorization
normal-series-snd-to-last unfolding  $G'$ -def by auto
  from  $G'G \ J(1,4)$  have  $G'J:G' \triangleleft (G \setminus \text{carrier} := J)$  by (metis normal-imp-subgroup
normal-restrict-supergroup)
  from finite  $J(1)$  have  $\text{fin}J:\text{finite } J$  by (auto simp: normal-imp-subgroup subgroup-finite)
  from  $JG'GG'$  simple-last have  $\text{rcosets}_{G \setminus \text{carrier} := J} G' = \{\mathbf{1}_{G \text{ Mod } G'}\} \vee$ 
 $\text{rcosets}_{G \setminus \text{carrier} := J} G' = \text{carrier } (G \text{ Mod } G')$ 
  unfolding simple-group-def simple-group-axioms-def by auto
  thus False
proof
  assume  $\text{rcosets}_{G \setminus \text{carrier} := J} G' = \{\mathbf{1}_{G \text{ Mod } G'}\}$ 
  hence  $\text{rcosets}_{G \setminus \text{carrier} := J} G' = \{\mathbf{1}_{(G \setminus \text{carrier} := J) \text{ Mod } G'}\}$  unfolding
FactGroup-def by simp
  hence  $G' = J$  using  $G'J \ \text{fin}J$  normal.fact-group-trivial-iff unfolding Fact-
Group-def by fastforce
  with  $J(2)$  show False by simp
next
  assume facts-eq: $\text{rcosets}_{G \setminus \text{carrier} := J} G' = \text{carrier } (G \text{ Mod } G')$ 
  have  $J = \text{carrier } G$ 
  proof
  show  $J \subseteq \text{carrier } G$  using  $J(1)$  normal-imp-subgroup subgroup.subset by
force
next
  show  $\text{carrier } G \subseteq J$ 
  proof
  fix  $x$ 
  assume  $x:x \in \text{carrier } G$ 
  hence  $G' \#> x \in \text{carrier } (G \text{ Mod } G')$  unfolding FactGroup-def

```

```

RCOSETS-def by auto
  hence  $G' \#> x \in \text{rcosets}_G(\text{carrier} := J)$   $G'$  using facts-eq by auto
  then obtain  $j$  where  $j:j \in J$   $G' \#> x = G' \#> j$  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  $g':g' \in G'$   $x = g' \otimes j$  unfolding r-coset-def by
auto
  hence  $g' \in J$  using  $G'J$  normal-imp-subgroup subgroup.subset by force
  with  $g'(2)$   $j(1)$  show  $x \in J$  using  $J(1)$  normal-imp-subgroup sub-
group.m-closed by fastforce
  qed
  qed
  with  $J(3)$  show False by simp
  qed
}
qed

```

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

```

lemma remdups-adj-obtain-adjacency:
  assumes  $i + 1 < \text{length} (\text{remdups-adj } xs)$   $\text{length } xs > 0$ 
  obtains  $j$  where  $j + 1 < \text{length } xs$ 
     $(\text{remdups-adj } xs) ! i = xs ! j$   $(\text{remdups-adj } xs) ! (i + 1) = xs ! (j + 1)$ 
using assms proof (induction  $xs$  arbitrary:  $i$  thesis)
  case Nil
  hence False by (metis length-greater-0-conv)
  thus thesis..
next
  case (Cons  $x$   $xs$ )
  then have  $xs \neq []$ 
  by auto
  then obtain  $y$   $xs'$  where  $xs: xs = y \# xs'$ 
  by (cases  $xs$ ) blast
  from  $\langle xs \neq [] \rangle$  have  $\text{len}xs:\text{length } xs > 0$  by simp
  from  $xs$  have  $\text{rem}:\text{remdups-adj} (x \# xs) = (\text{if } x = y \text{ then } \text{remdups-adj} (y \# xs')
\text{ else } x \# \text{remdups-adj} (y \# xs'))$  using remdups-adj.simps(3) by auto
  show thesis
  proof (cases  $x = y$ )
  case True
  with  $\text{rem } xs$  have  $\text{rem2}:\text{remdups-adj} (x \# xs) = \text{remdups-adj } xs$  by auto
  with Cons(3) have  $i + 1 < \text{length} (\text{remdups-adj } xs)$  by simp
  with Cons.IH  $\text{len}xs$  obtain  $k$  where  $j:k + 1 < \text{length } xs$   $\text{remdups-adj } xs ! i =
xs ! k$ 
     $\text{remdups-adj } xs ! (i + 1) = xs ! (k + 1)$  by auto
  thus thesis using Cons(2)  $\text{rem2}$  by auto
  case False
  with  $\text{rem } xs$  have  $\text{rem2}:\text{remdups-adj} (x \# xs) = x \# \text{remdups-adj } xs$  by auto
  show thesis

```

```

proof (cases i)
  case 0
    have  $0 + 1 < \text{length } (x \# xs)$  using lenxs by auto
    moreover have  $\text{remdups-adj } (x \# xs) ! i = (x \# xs) ! 0$ 
    proof -
      have  $\text{remdups-adj } (x \# xs) ! i = (x \# \text{remdups-adj } (y \# xs')) ! 0$  using xs
rem2 0 by simp
      also have  $\dots = x$  by simp
      also have  $\dots = (x \# xs) ! 0$  by simp
      finally show ?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 xs rem2 0 by simp
      also have  $\dots = \text{remdups-adj } (y \# xs') ! 0$  by simp
      also have  $\dots = (y \# (\text{remdups } (y \# xs'))) ! 0$  by (metis nth-Cons'
remdups-adj-Cons-alt)
      also have  $\dots = y$  by simp
      also have  $\dots = (x \# xs) ! (0 + 1)$  unfolding xs by simp
      finally show ?thesis.
    qed
    ultimately show thesis by (rule Cons.prem1)
  next
    case (Suc k)
    with Cons3 have  $k + 1 < \text{length } (\text{remdups-adj } (x \# xs)) - 1$  by auto
    also have  $\dots \leq \text{length } (\text{remdups-adj } xs) + 1 - 1$  by (metis One-nat-def
le-refl list.size4) rem2)
    also have  $\dots = \text{length } (\text{remdups-adj } xs)$  by simp
    finally have  $k + 1 < \text{length } (\text{remdups-adj } xs)$ .
    with Cons.IH lenxs obtain j where  $j : j + 1 < \text{length } xs$   $\text{remdups-adj } xs ! k$ 
     $= xs ! j$ 
    remdups-adj } xs ! (k + 1) = xs ! (j + 1) by auto
    from j1 have  $\text{Suc } j + 1 < \text{length } (x \# xs)$  by simp
    moreover have  $\text{remdups-adj } (x \# xs) ! i = (x \# xs) ! (\text{Suc } j)$ 
    proof -
      have  $\text{remdups-adj } (x \# xs) ! i = (x \# \text{remdups-adj } xs) ! i$  using rem2 by
simp
      also have  $\dots = (\text{remdups-adj } xs) ! k$  using Suc by simp
      also have  $\dots = xs ! j$  using j2.
      also have  $\dots = (x \# xs) ! (\text{Suc } j)$  by simp
      finally show ?thesis.
    qed
    moreover have  $\text{remdups-adj } (x \# xs) ! (i + 1) = (x \# xs) ! (\text{Suc } j + 1)$ 
    proof -
      have  $\text{remdups-adj } (x \# xs) ! (i + 1) = (x \# \text{remdups-adj } xs) ! (i + 1)$ 
using rem2 by simp
      also have  $\dots = (\text{remdups-adj } xs) ! (k + 1)$  using Suc by simp
      also have  $\dots = xs ! (j + 1)$  using j3.

```

also have $\dots = (x \# xs) ! (Suc\ j + 1)$ **by** *simp*
finally show *?thesis*.
qed
ultimately show *thesis* **by** (*rule Cons.prem1*)
qed
qed
qed

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

lemma *remdups-adj-adjacent*:
 $Suc\ i < length\ (remdups\ adj\ xs) \implies remdups\ adj\ xs\ !\ i \neq remdups\ adj\ xs\ !\ Suc\ i$
proof (*induction xs arbitrary: i rule: remdups-adj.induct*)
case ($\exists\ x\ y\ xs\ i$)
thus *?case* **by** (*cases i, cases x = y*) (*simp, auto simp: hd-conv-nth[symmetric]*)
qed *simp-all*

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

lemma (*in composition-series*) *intersect-normal*:
assumes *finite:finite* (*carrier G*)
assumes *KG:K* $\triangleleft G$
shows *composition-series* ($G \langle carrier := K \rangle$) (*remdups-adj* (*map* ($\lambda H. K \cap H$) \mathfrak{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* ($G \langle carrier := K \rangle$) **using** *KG normal-imp-subgroup subgroup-imp-group*
by *auto*
next
 – Show, that removing adjacent duplicates doesn't result in an empty list.
assume $remdups\ adj\ (map\ ((\cap)\ K)\ \mathfrak{G}) = []$
hence $map\ ((\cap)\ K)\ \mathfrak{G} = []$ **by** (*metis remdups-adj-Nil-iff*)
hence $\mathfrak{G} = []$ **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 $\mathfrak{G} = \{1\} \# tl\ \mathfrak{G}$ **using** *notempty hd* **by** (*metis list.sel(1,3) neq-Nil-conv*)
hence $map\ ((\cap)\ K)\ \mathfrak{G} = map\ ((\cap)\ K)\ (\{1\} \# tl\ \mathfrak{G})$ **by** *simp*
hence $remdups\ adj\ (map\ ((\cap)\ K)\ \mathfrak{G}) = remdups\ adj\ ((K \cap \{1\}) \# (map\ ((\cap)\ K)\ (tl\ \mathfrak{G})))$ **by** *simp*
also have $\dots = (K \cap \{1\}) \# tl\ (remdups\ adj\ ((K \cap \{1\}) \# (map\ ((\cap)\ K)\ (tl\ \mathfrak{G}))))$ **by** *simp*
finally have $hd\ (remdups\ adj\ (map\ ((\cap)\ K)\ \mathfrak{G})) = K \cap \{1\}$ **using** *list.sel(1)*
by *metis*
thus $hd\ (remdups\ adj\ (map\ ((\cap)\ K)\ \mathfrak{G})) = \{1_{G \langle carrier := K \rangle}\}$
using *KG normal-imp-subgroup subgroup.one-closed* **by** *force*

next

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

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

hence $rev (map ((\cap) K) \mathfrak{G}) = map ((\cap) K) ((carrier\ G) \# tl (rev \mathfrak{G}))$ **by** $(metis\ rev-map)$

hence $rev:rev (map ((\cap) K) \mathfrak{G}) = (K \cap (carrier\ G)) \# (map ((\cap) K) (tl (rev \mathfrak{G})))$ **by** $simp$

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

by $(metis\ hd-rev\ map-is-Nil-conv\ notempty\ remdups-adj-Nil-iff)$

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

also have $\dots = hd (remdups-adj ((K \cap (carrier\ G)) \# (map ((\cap) K) (tl (rev \mathfrak{G}))))$ **by** $(metis\ rev)$

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

also have $\dots = K$ **using** $KG\ normal-imp-subgroup\ subgroup.subset$ **by** $force$

finally show $last (remdups-adj (map ((\cap) K) \mathfrak{G})) = carrier (G \langle carrier := K \rangle)$ **by** $auto$

next

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

fix j

assume $j:j + 1 < length (remdups-adj (map ((\cap) K) \mathfrak{G}))$

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

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

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

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

using $remdups-adj-obtain-adjacency$ **by** $force$

from $i(1)$ **have** $i':i + 1 < length \mathfrak{G}$ **by** $(metis\ length-map)$

hence $GiSi:\mathfrak{G} ! i \triangleleft G \langle carrier := \mathfrak{G} ! (i + 1) \rangle$ **by** $(metis\ normal)$

hence $GiSi':\mathfrak{G} ! i \subseteq \mathfrak{G} ! (i + 1)$ **using** $normal-imp-subgroup\ subgroup.subset$ **by** $force$

from i' **have** $finGSi:finite (\mathfrak{G} ! (i + 1))$ **using** $normal-series-subgroups\ finite$ **by** $(metis\ subgroup-finite)$

from $GiSi\ KG\ i'$ $normal-series-subgroups$ **have** $GSiK\ normGSi:\mathfrak{G} ! (i + 1) \cap K \triangleleft G \langle carrier := \mathfrak{G} ! (i + 1) \rangle$

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

unfolding $second-isomorphism-grp-def\ second-isomorphism-grp-axioms-def$ **by** $auto$

with $GiSi$ **have** $\mathfrak{G} ! i \cap (\mathfrak{G} ! (i + 1) \cap K) \triangleleft G \langle carrier := \mathfrak{G} ! (i + 1) \rangle$

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

hence $K \cap (\mathfrak{G} ! i \cap \mathfrak{G} ! (i + 1)) \triangleleft G \langle carrier := \mathfrak{G} ! (i + 1) \rangle$ **by** $(metis\ inf-commute\ inf-left-commute)$

hence $KG\ inormGSi:K \cap \mathfrak{G} ! i \triangleleft G \langle carrier := \mathfrak{G} ! (i + 1) \rangle$ **using** $GiSi'$ **by** $(metis\ le-iff-inf)$

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

moreover have $groupGSi:group (G \langle carrier := \mathfrak{G} ! (i + 1) \rangle)$ **using** $i\ nor-$

mal-series-subgroups subgroup-imp-group by auto
moreover have *subKGSiGSi:subgroup* ($K \cap \mathfrak{G}!(i+1)$) ($G(\text{carrier} := \mathfrak{G}!(i+1))$) **by** (*metis GSiKnormGSi inf-sup-aci(1) normal-imp-subgroup*)
ultimately have *fstgoal*: $K \cap \mathfrak{G}!i \triangleleft G(\text{carrier} := \mathfrak{G}!(i+1), \text{carrier} := K \cap \mathfrak{G}!(i+1))$
using *group.normal-restrict-supergroup by force*
thus *remdups-adj* ($\text{map}((\cap) K) \mathfrak{G}!j \triangleleft G(\text{carrier} := K, \text{carrier} := \text{remdups-adj}(\text{map}((\cap) K) \mathfrak{G}!(j+1)))$)
using *i by auto*
from *simplefact* **have** *Gisimple:simple-group* ($G(\text{carrier} := \mathfrak{G}!(i+1)) \text{Mod } \mathfrak{G}!i$) **using** *i' by simp*
hence *Gimax:max-normal-subgroup* ($\mathfrak{G}!i$) ($G(\text{carrier} := \mathfrak{G}!(i+1))$)
using *normal.max-normal-simple-quotient GiSi finGSi by force*
from *GSiKnormGSi GiSi* **have** $\mathfrak{G}!i \triangleleft_{\#} G(\text{carrier} := \mathfrak{G}!(i+1)) \mathfrak{G}!(i+1) \cap K \triangleleft (G(\text{carrier} := \mathfrak{G}!(i+1)))$
using *groupGSi group.normal-subgroup-set-mult-closed set-mult-consistent by fastforce*
hence $\mathfrak{G}!i \triangleleft_{\#} \mathfrak{G}!(i+1) \cap K \triangleleft G(\text{carrier} := \mathfrak{G}!(i+1))$ **unfolding** *set-mult-def by auto*
hence $\mathfrak{G}!i \triangleleft_{\#} K \cap \mathfrak{G}!(i+1) \triangleleft G(\text{carrier} := \mathfrak{G}!(i+1))$ **using** *inf-commute by metis*
moreover have $\mathfrak{G}!i \subseteq \mathfrak{G}!i \triangleleft_{\#} G(\text{carrier} := \mathfrak{G}!(i+1)) K \cap \mathfrak{G}!(i+1)$
using *second-isomorphism-grp.H-contained-in-set-mult*
unfolding *second-isomorphism-grp-def second-isomorphism-grp-axioms-def*
using *subKGSiGSi GiSi normal-imp-subgroup by fastforce*
hence $\mathfrak{G}!i \subseteq \mathfrak{G}!i \triangleleft_{\#} K \cap \mathfrak{G}!(i+1)$ **unfolding** *set-mult-def by auto*
ultimately have *KGdisj*: $\mathfrak{G}!i \triangleleft_{\#} K \cap \mathfrak{G}!(i+1) = \mathfrak{G}!i \vee \mathfrak{G}!i \triangleleft_{\#} K \cap \mathfrak{G}!(i+1) = \mathfrak{G}!(i+1)$
using *Gimax unfolding max-normal-subgroup-def max-normal-subgroup-axioms-def by auto*
obtain φ **where** $\varphi \in \text{iso} (G(\text{carrier} := K \cap \mathfrak{G}!(i+1)) \text{Mod} (\mathfrak{G}!i \cap (K \cap \mathfrak{G}!(i+1))))$
 $(G(\text{carrier} := \mathfrak{G}!i \triangleleft_{\#} G(\text{carrier} := \mathfrak{G}!(i+1)) K \cap \mathfrak{G}!(i+1)) \text{Mod } \mathfrak{G}!i)$
using *second-isomorphism-grp.normal-intersection-quotient-isom*
unfolding *second-isomorphism-grp-def second-isomorphism-grp-axioms-def*
using *GiSi subKGSiGSi normal-imp-subgroup by fastforce*
hence $\varphi \in \text{iso} (G(\text{carrier} := K \cap \mathfrak{G}!(i+1)) \text{Mod} (K \cap \mathfrak{G}!(i+1) \cap \mathfrak{G}!i))$
 $(G(\text{carrier} := \mathfrak{G}!i \triangleleft_{\#} G(\text{carrier} := \mathfrak{G}!(i+1)) K \cap \mathfrak{G}!(i+1)) \text{Mod } \mathfrak{G}!i)$
by (*metis inf-commute*)
hence $\varphi \in \text{iso} (G(\text{carrier} := K \cap \mathfrak{G}!(i+1)) \text{Mod} (K \cap (\mathfrak{G}!(i+1) \cap \mathfrak{G}!i)))$
 $(G(\text{carrier} := \mathfrak{G}!i \triangleleft_{\#} G(\text{carrier} := \mathfrak{G}!(i+1)) K \cap \mathfrak{G}!(i+1)) \text{Mod } \mathfrak{G}!i)$
by (*metis Int-assoc*)
hence $\varphi \in \text{iso} (G(\text{carrier} := K \cap \mathfrak{G}!(i+1)) \text{Mod} (K \cap \mathfrak{G}!i))$
 $(G(\text{carrier} := \mathfrak{G}!i \triangleleft_{\#} G(\text{carrier} := \mathfrak{G}!(i+1)) K \cap \mathfrak{G}!(i+1))$

Mod $\mathfrak{G} ! i$
by (*metis GiSi' Int-absorb2 Int-commute*)
hence $\varphi: \varphi \in \text{iso } (G(\text{carrier} := K \cap \mathfrak{G} ! (i + 1))) \text{ Mod } (K \cap \mathfrak{G} ! i)$
 $(G(\text{carrier} := \mathfrak{G} ! i <\#\#> K \cap \mathfrak{G} ! (i + 1))) \text{ Mod } \mathfrak{G} ! i$
unfolding *set-mult-def* **by** *auto*
from *fstgoal* **have** *KGsiKGigroup:group* $(G(\text{carrier} := K \cap \mathfrak{G} ! (i + 1))) \text{ Mod } (K \cap \mathfrak{G} ! i)$ **using** *normal.factorgroup-is-group* **by** *auto*
from *KGdisj* **show** *simple-group* $(G(\text{carrier} := K, \text{carrier} := \text{remdups-adj } (\text{map } ((\cap) K) \mathfrak{G}) ! j)) \text{ Mod } \text{remdups-adj } (\text{map } ((\cap) K) \mathfrak{G}) ! j$
proof *auto*
have *groupGi:group* $(G(\text{carrier} := \mathfrak{G} ! i))$ **using** *i' normal-series-subgroups subgroup-imp-group* **by** *auto*
assume $\mathfrak{G} ! i <\#\#> K \cap \mathfrak{G} ! \text{Suc } i = \mathfrak{G} ! i$
with φ **have** $\varphi \in \text{iso } (G(\text{carrier} := K \cap \mathfrak{G} ! (i + 1))) \text{ Mod } (K \cap \mathfrak{G} ! i)$
 $(G(\text{carrier} := \mathfrak{G} ! i)) \text{ Mod } \mathfrak{G} ! i$ **by** *auto*
moreover **obtain** ψ **where** $\psi \in \text{iso } (G(\text{carrier} := \mathfrak{G} ! i)) \text{ Mod } (\text{carrier } (G(\text{carrier} := \mathfrak{G} ! i))))$
 $(G(\text{carrier} := \{1\})) (G(\text{carrier} := \{1_{G(\text{carrier} := \mathfrak{G} ! i)}\}))$
using *group.self-factor-iso groupGi* **by** *force*
ultimately **obtain** π **where** $\pi \in \text{iso } (G(\text{carrier} := K \cap \mathfrak{G} ! (i + 1))) \text{ Mod } (K \cap \mathfrak{G} ! i)$
 $(G(\text{carrier} := \{1\}))$
using *iso-set-trans* **by** *fastforce*
hence $\text{order } (G(\text{carrier} := K \cap \mathfrak{G} ! (i + 1))) \text{ Mod } (K \cap \mathfrak{G} ! i) = \text{order } (G(\text{carrier} := \{1\}))$ **by** (*metis iso-order-closed*)
hence $\text{order } (G(\text{carrier} := K \cap \mathfrak{G} ! (i + 1))) \text{ Mod } (K \cap \mathfrak{G} ! i) = 1$ **unfolding** *order-def* **by** *auto*
hence $\text{carrier } (G(\text{carrier} := K \cap \mathfrak{G} ! (i + 1))) \text{ Mod } (K \cap \mathfrak{G} ! i) = \{1_{G(\text{carrier} := K \cap \mathfrak{G} ! (i + 1))} \text{ Mod } (K \cap \mathfrak{G} ! i)}\}$
using *group.order-one-triv-iff KGsiKGigroup* **by** *blast*
moreover **from** *fstgoal* **have** $K \cap \mathfrak{G} ! i \triangleleft G(\text{carrier} := K \cap \mathfrak{G} ! (i + 1))$ **by** *auto*
moreover **from** *finGSi* **have** *finite* $(\text{carrier } (G(\text{carrier} := K \cap \mathfrak{G} ! (i + 1))))$ **by** *auto*
ultimately **have** $K \cap \mathfrak{G} ! i = \text{carrier } (G(\text{carrier} := K \cap \mathfrak{G} ! (i + 1)))$ **by** (*metis normal.fact-group-trivial-iff*)
hence $(\text{remdups-adj } (\text{map } ((\cap) K) \mathfrak{G})) ! j = (\text{remdups-adj } (\text{map } ((\cap) K) \mathfrak{G})) ! (j + 1)$ **using** *i* **by** *auto*
with *j* **have** *False* **using** *remdups-adj-adjacent KGnotempty Suc-eq-plus1* **by** *metis*
thus *simple-group* $(G(\text{carrier} := \text{remdups-adj } (\text{map } ((\cap) K) \mathfrak{G}) ! \text{Suc } j)) \text{ Mod } \text{remdups-adj } (\text{map } ((\cap) K) \mathfrak{G}) ! j$
next
assume $\mathfrak{G} ! i <\#\#> K \cap \mathfrak{G} ! \text{Suc } i = \mathfrak{G} ! \text{Suc } i$
moreover **with** φ **have** $\varphi \in \text{iso } (G(\text{carrier} := K \cap \mathfrak{G} ! (i + 1))) \text{ Mod } (K \cap \mathfrak{G} ! i)$
 $(G(\text{carrier} := \mathfrak{G} ! (i + 1))) \text{ Mod } \mathfrak{G} ! i$ **by** *auto*
then **obtain** φ' **where** $\varphi' \in \text{iso } (G(\text{carrier} := \mathfrak{G} ! (i + 1))) \text{ Mod } \mathfrak{G} ! i$
 $(G(\text{carrier} := K \cap \mathfrak{G} ! (i + 1))) \text{ Mod } (K \cap \mathfrak{G} ! i)$
using *KGsiKGigroup group.iso-set-sym* **by** *auto*
with *Gisimple KGsiKGigroup* **have** *simple-group* $(G(\text{carrier} := K \cap \mathfrak{G} ! (i + 1))) \text{ Mod } (K \cap \mathfrak{G} ! i)$ **by** (*metis simple-group.iso-simple*)
with *i* **show** *simple-group* $(G(\text{carrier} := \text{remdups-adj } (\text{map } ((\cap) K) \mathfrak{G}) ! \text{Suc } i)) \text{ Mod } \text{remdups-adj } (\text{map } ((\cap) K) \mathfrak{G}) ! \text{Suc } i$

j) *Mod* *remdups-adj* (*map* ((\cap) K) \mathfrak{G}) ! j) **by** *auto*
qed
qed

lemma (*in group*) *composition-series-extend*:
assumes *composition-series* (G (*carrier* := H)) \mathfrak{H}
assumes *simple-group* (G *Mod* H) $H \triangleleft G$
shows *composition-series* G (\mathfrak{H} @ [*carrier* G])
unfolding *composition-series-def* *composition-series-axioms-def*
proof *auto*
from *assms*(1) **interpret** *comp* \mathfrak{H} : *composition-series* G (*carrier* := H) \mathfrak{H} .
show *normal-series* G (\mathfrak{H} @ [*carrier* G]) **using** *assms*(3) *comp* \mathfrak{H} .*is-normal-series*
by (*metis normal-series-extend*)
fix i
assume $i < \text{length } \mathfrak{H}$
show *simple-group* (G (*carrier* := (\mathfrak{H} @ [*carrier* G]) ! *Suc* i) *Mod* (\mathfrak{H} @ [*carrier* G]) ! i)
proof (*cases* $i = \text{length } \mathfrak{H} - 1$)
case *True*
hence (\mathfrak{H} @ [*carrier* G]) ! *Suc* $i = \text{carrier } G$ **by** (*metis* i *diff-Suc-1* *lessE* *nth-append-length*)
moreover **have** (\mathfrak{H} @ [*carrier* G]) ! $i = \mathfrak{H}$! i **by** (*metis* *butlast-snoc* i *nth-butlast*)
hence (\mathfrak{H} @ [*carrier* G]) ! $i = H$ **using** *True* *last-conv-nth* *comp* \mathfrak{H} .*notempty* *comp* \mathfrak{H} .*last* **by** *auto*
ultimately **show** *?thesis* **using** *assms*(2) **by** *auto*
next
case *False*
hence *Suc* $i < \text{length } \mathfrak{H}$ **using** i **by** *auto*
hence (\mathfrak{H} @ [*carrier* G]) ! *Suc* $i = \mathfrak{H}$! *Suc* i **using** *nth-append* **by** *metis*
moreover **from** i **have** (\mathfrak{H} @ [*carrier* G]) ! $i = \mathfrak{H}$! i **using** *nth-append* **by** *metis*
ultimately **show** *?thesis* **using** $\langle \text{Suc } i < \text{length } \mathfrak{H} \rangle$ *comp* \mathfrak{H} .*simplefact* **by** *auto*
qed
qed

lemma (*in composition-series*) *entries-mono*:
assumes $i \leq j < \text{length } \mathfrak{G}$
shows \mathfrak{G} ! $i \subseteq \mathfrak{G}$! j
using *assms* **proof** (*induction* $j - i$ *arbitrary: i j*)
case 0
hence $i = j$ **by** *auto*
thus \mathfrak{G} ! $i \subseteq \mathfrak{G}$! j **by** *auto*
next
case (*Suc* k i j)
hence $i' : i + (\text{Suc } k) = j$ $i + 1 < \text{length } \mathfrak{G}$ **by** *auto*
hence $i' : i + 1 \leq j$ **by** *auto*
have \mathfrak{G} ! $i \subseteq \mathfrak{G}$! ($i + 1$) **using** i' *normal* *normal-imp-subgroup* *subgroup.subset*
by *force*
moreover **have** $j - (i + 1) = k$ $j < \text{length } \mathfrak{G}$ **using** *Suc* *assms* **by** *auto*

```

    hence  $\mathfrak{G} ! (i + 1) \subseteq \mathfrak{G} ! j$  using Suc(1) ij by auto
    ultimately show  $\mathfrak{G} ! i \subseteq \mathfrak{G} ! j$  by 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  $\bigwedge a. (\text{carrier} = \{a\}, \text{mult} = (\lambda x y. x), \text{one} = a) \in \{G. \text{group } G\}$ 
  unfolding group-def group-axioms-def monoid-def Units-def by auto
qed

```

```

definition group-iso-rel :: 'a group  $\Rightarrow$  'a group  $\Rightarrow$  bool
  where group-iso-rel G H =  $(\exists \varphi. \varphi \in \text{iso } (\text{Rep-group } G) (\text{Rep-group } H))$ 

```

```

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 sympI)
    fix G H :: 'b group
    assume group-iso-rel G H
    then obtain  $\varphi$  where  $\varphi \in \text{iso } (\text{Rep-group } G) (\text{Rep-group } H)$  unfolding
group-iso-rel-def by auto
    then obtain  $\varphi'$  where  $\varphi' \in \text{iso } (\text{Rep-group } H) (\text{Rep-group } G)$  using group.iso-sym
Rep-group
    using group.iso-set-sym by blast
    thus group-iso-rel H G unfolding group-iso-rel-def by auto
  qed
next
  show transp group-iso-rel
  proof (rule transpI)

```

```

fix  $G H I :: 'b$  group
assume  $group\text{-}iso\text{-}rel\ G\ H\ group\text{-}iso\text{-}rel\ H\ I$ 
then obtain  $\varphi\ \psi$  where  $\varphi \in iso\ (Rep\text{-}group\ G)\ (Rep\text{-}group\ H)$   $\psi \in iso$ 
 $(Rep\text{-}group\ H)\ (Rep\text{-}group\ I)$ 
unfolding  $group\text{-}iso\text{-}rel\text{-}def$  by  $auto$ 
then obtain  $\pi$  where  $\pi \in iso\ (Rep\text{-}group\ G)\ (Rep\text{-}group\ I)$ 
using  $iso\text{-}set\text{-}trans$  by  $blast$ 
thus  $group\text{-}iso\text{-}rel\ G\ I$  unfolding  $group\text{-}iso\text{-}rel\text{-}def$  by  $auto$ 
qed
qed

```

This assigns to a given group the group isomorphism class

```

definition (in  $group$ )  $iso\text{-}class :: 'a$  group-iso-class
where  $iso\text{-}class = Abs\text{-}group\text{-}iso\ (Abs\text{-}group\ (monoid.truncate\ G))$ 

```

Two isomorphic groups do indeed have the same isomorphism class:

lemma $iso\text{-}classes\text{-}iff$:

```

assumes  $group\ G$ 
assumes  $group\ H$ 
shows  $(\exists \varphi. \varphi \in iso\ G\ H) = (group.iso\text{-}class\ G = group.iso\text{-}class\ H)$ 
proof –
from  $assms(1,2)$  have  $groups:group\ (monoid.truncate\ G)\ group\ (monoid.truncate$ 
 $H)$ 
unfolding  $monoid.truncate\text{-}def\ group\text{-}def\ group\text{-}axioms\text{-}def\ Units\text{-}def\ monoid\text{-}def$ 
by  $auto$ 
have  $(\exists \varphi. \varphi \in iso\ G\ H) = (\exists \varphi. \varphi \in iso\ (monoid.truncate\ G)\ (monoid.truncate$ 
 $H))$ 
unfolding  $iso\text{-}def\ hom\text{-}def\ monoid.truncate\text{-}def$  by  $auto$ 
also have  $\dots = group\text{-}iso\text{-}rel\ (Abs\text{-}group\ (monoid.truncate\ G))\ (Abs\text{-}group\ (monoid.truncate$ 
 $H))$ 
unfolding  $group\text{-}iso\text{-}rel\text{-}def$  using  $groups\ group.Abs\text{-}group\text{-}inverse$  by  $(metis$ 
 $mem\text{-}Collect\text{-}eq)$ 
also have  $\dots = (group.iso\text{-}class\ G = group.iso\text{-}class\ H)$  using  $group.iso\text{-}class\text{-}def$ 
 $assms\ group\text{-}iso\text{-}class.abs\text{-}eq\text{-}iff$  by  $metis$ 
finally show  $?thesis$ .
qed

```

end

theory $JordanHolder$

imports

```

 $CompositionSeries$ 
 $MaximalNormalSubgroups$ 
 $HOL\text{-}Library.Multiset$ 
 $GroupIsoClasses$ 

```

begin

10 The Jordan-Hölder Theorem

locale *jordan-hoelder* = group
 + *comp* \mathfrak{H} ?: composition-series $G \mathfrak{H}$
 + *comp* \mathfrak{G} ?: composition-series $G \mathfrak{G}$ for \mathfrak{H} and \mathfrak{G}
 + **assumes** *finite:finite* (carrier G)

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

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

assumes *length* $\mathfrak{G} > 1$
shows *butlast quotients* = *normal-series.quotients* ($G(\downarrow \text{carrier} := \mathfrak{G} ! (\text{length } \mathfrak{G} - 1 - 1))$) (*take* (*length* $\mathfrak{G} - 1$) \mathfrak{G})
proof (*rule nth-equalityI*)
define n **where** $n = \text{length } \mathfrak{G} - 1$
hence $n = \text{length} (\text{take } n \mathfrak{G})$ $n > 0$ $n < \text{length } \mathfrak{G}$ **using** *assms notempty* **by** *auto*
interpret *normal* \mathfrak{G} *butlast*: *normal-series* ($G(\downarrow \text{carrier} := \mathfrak{G} ! (n - 1))$) *take* $n \mathfrak{G}$

using *normal-series-prefix-closed* $\langle n > 0 \rangle \langle n < \text{length } \mathfrak{G} \rangle$ **by** *auto*
have *length* (*butlast quotients*) = *length quotients* - 1 **by** (*metis length-butlast*)
also have ... = *length* $\mathfrak{G} - 1 - 1$ **by** (*metis add-diff-cancel-right' quotients-length*)
also have ... = *length* (*take* $n \mathfrak{G}$) - 1 **by** (*metis* $\langle n = \text{length} (\text{take } n \mathfrak{G}) \rangle$ *n-def*)
also have ... = *length normal* \mathfrak{G} *butlast.quotients* **by** (*metis normal* \mathfrak{G} *butlast.quotients-length diff-add-inverse2*)
finally show *length* (*butlast quotients*) = *length normal* \mathfrak{G} *butlast.quotients* .
have $\forall i < \text{length} (\text{butlast quotients}). \text{butlast quotients} ! i = \text{normal}\mathfrak{G}\text{butlast.quotients} ! i$
proof *auto*
fix i
assume $i : i < \text{length quotients} - \text{Suc } 0$
hence $i' : i < \text{length } \mathfrak{G} - 1$ $i < n$ $i + 1 < n$ **unfolding** *n-def* **using** *quotients-length* **by** *auto*
from i **have** *butlast quotients* ! $i = \text{quotients} ! i$ **by** (*metis One-nat-def length-butlast nth-butlast*)
also have ... = $G(\downarrow \text{carrier} := \mathfrak{G} ! (i + 1)) \text{Mod } \mathfrak{G} ! i$ **unfolding** *quotients-def* **using** $i'(1)$ **by** *auto*
also have ... = $G(\downarrow \text{carrier} := (\text{take } n \mathfrak{G}) ! (i + 1)) \text{Mod } (\text{take } n \mathfrak{G}) ! i$ **using** $i'(2,3)$ *nth-take* **by** *metis*
also have ... = *normal* \mathfrak{G} *butlast.quotients* ! i **unfolding** *normal* \mathfrak{G} *butlast.quotients-def* **using** i' **by** *fastforce*
finally show *butlast* (*normal-series.quotients* $G \mathfrak{G}$) ! $i = \text{normal-series.quotients} (G(\downarrow \text{carrier} := \mathfrak{G} ! (n - \text{Suc } 0))) (\text{take } n \mathfrak{G}) ! i$ **by** *auto*
qed
thus $\bigwedge i. i < \text{length} (\text{butlast quotients}) \implies \text{butlast quotients} ! i = \text{normal-series.quotients} (G(\downarrow \text{carrier} := \mathfrak{G} ! (\text{length } \mathfrak{G} - 1 - 1))) (\text{take } (\text{length } \mathfrak{G} - 1) \mathfrak{G}) ! i$

unfolding *n-def* **by** *auto*
qed

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

theorem *jordan-hoelder-multisets*:

assumes *group* G

assumes *finite* (*carrier* G)

assumes *composition-series* $G \mathfrak{G}$

assumes *composition-series* $G \mathfrak{H}$

shows *mset* (*map* *group.iso-class* (*normal-series.quotients* $G \mathfrak{G}$))
 $=$ *mset* (*map* *group.iso-class* (*normal-series.quotients* $G \mathfrak{H}$))

using *assms*

proof (*induction length* \mathfrak{G} *arbitrary*: $\mathfrak{G} \mathfrak{H} G$ *rule*: *full-nat-induct*)

case ($1 \mathfrak{G} \mathfrak{H} G$)

then interpret *comp* \mathfrak{G} : *composition-series* $G \mathfrak{G}$ **by** *simp*

from 1 **interpret** *comp* \mathfrak{H} : *composition-series* $G \mathfrak{H}$ **by** *simp*

from 1 **interpret** *grp* G : *group* G **by** *simp*

show *?case*

proof (*cases length* $\mathfrak{G} \leq 2$)

next

case *True*

hence *length* $\mathfrak{G} = 0 \vee$ *length* $\mathfrak{G} = 1 \vee$ *length* $\mathfrak{G} = 2$ **by** *arith*

with *comp* \mathfrak{G} .*notempty* **have** *length* $\mathfrak{G} = 1 \vee$ *length* $\mathfrak{G} = 2$ **by** *simp*

thus *?thesis*

proof (*auto simp del*: *mset-map*)

— First trivial case: \mathfrak{G} is the trivial group.

assume *length* $\mathfrak{G} =$ *Suc* 0

hence *length*:*length* $\mathfrak{G} = 1$ **by** *simp*

hence *length* $[] + 1 =$ *length* \mathfrak{G} **by** *auto*

moreover from *length* **have** *char* \mathfrak{G} : $\mathfrak{G} = [\{1_G\}]$ **by** (*metis comp* \mathfrak{G} .*composition-series-length-one*)

hence *carrier* $G = \{1_G\}$ **by** (*metis comp* \mathfrak{G} .*composition-series-triv-group*)

with *length char* \mathfrak{G} **have** $\mathfrak{G} = \mathfrak{H}$ **using** *comp* \mathfrak{H} .*composition-series-triv-group*

by *simp*

thus *?thesis* **by** *simp*

next

— Second trivial case: \mathfrak{G} is simple.

assume *length* $\mathfrak{G} = 2$

hence \mathfrak{G} *char*: $\mathfrak{G} = [\{1_G\},$ *carrier* $G]$ **by** (*metis comp* \mathfrak{G} .*length-two-unique*)

hence *simple:simple-group* G **by** (*metis comp* \mathfrak{G} .*composition-series-simple-group*)

hence $\mathfrak{H} = [\{1_G\},$ *carrier* $G]$ **using** *comp* \mathfrak{H} .*composition-series-simple-group*

by *auto*

with \mathfrak{G} *char* **have** $\mathfrak{G} = \mathfrak{H}$ **by** *simp*

thus *?thesis* **by** *simp*

qed

next

case *False*

— Non-trivial case: \mathfrak{G} has length at least 3.
hence $\text{length:length } \mathfrak{G} \geq 3$ **by** *simp*
 — First we show that \mathfrak{H} must have a length of at least 3.
hence \neg *simple-group* G **using** $\text{comp}\mathfrak{G}.\text{composition-series-simple-group}$ **by** *auto*
hence $\mathfrak{H} \neq [\{1_G\}, \text{carrier } G]$ **using** $\text{comp}\mathfrak{H}.\text{composition-series-simple-group}$ **by**
auto
hence $\text{length } \mathfrak{H} \neq 2$ **using** $\text{comp}\mathfrak{H}.\text{length-two-unique}$ **by** *auto*
moreover from $\text{length have carrier } G \neq \{1_G\}$ **using** $\text{comp}\mathfrak{G}.\text{composition-series-length-one}$
 $\text{comp}\mathfrak{G}.\text{composition-series-triv-group}$ **by** *auto*
hence $\text{length } \mathfrak{H} \neq 1$ **using** $\text{comp}\mathfrak{H}.\text{composition-series-length-one}$ $\text{comp}\mathfrak{H}.\text{composition-series-triv-group}$
by *auto*
moreover from $\text{comp}\mathfrak{H}.\text{notempty}$ **have** $\text{length } \mathfrak{H} \neq 0$ **by** *simp*
ultimately have $\text{length}\mathfrak{H}\text{big:length } \mathfrak{H} \geq 3$ **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 $\text{length}\mathfrak{H}\text{big}$ **have** $m':m > 0 \ m < \text{length } \mathfrak{H} \ (m - 1) + 1 < \text{length } \mathfrak{H} \ m -$
 $1 = \text{length } \mathfrak{H} - 2 \ m - 1 + 1 = \text{length } \mathfrak{H} - 1 \ m - 1 < \text{length } \mathfrak{H}$
unfolding $m\text{-def}$ **by** *auto*
from length **have** $n':n > 0 \ n < \text{length } \mathfrak{G} \ (n - 1) + 1 < \text{length } \mathfrak{G} \ n - 1 <$
 $\text{length } \mathfrak{G} \ \text{Suc } n \leq \text{length } \mathfrak{G}$
 $n - 1 = \text{length } \mathfrak{G} - 2 \ n - 1 + 1 = \text{length } \mathfrak{G} - 1$ **unfolding** $n\text{-def}$ **by** *auto*
define $\mathfrak{G}Pn$ **where** $\mathfrak{G}Pn = G(\text{carrier} := \mathfrak{G} ! (n - 1))$
define $\mathfrak{H}Pm$ **where** $\mathfrak{H}Pm = G(\text{carrier} := \mathfrak{H} ! (m - 1))$
then interpret $\text{grp}\mathfrak{G}Pn$: *group* $\mathfrak{G}Pn$ **unfolding** $\mathfrak{G}Pn\text{-def}$ **using** n' **by** (*metis*
 $\text{comp}\mathfrak{G}.\text{normal-series-subgroups}$ $\text{comp}\mathfrak{G}.\text{subgroup-imp-group}$)
interpret $\text{grp}\mathfrak{H}Pm$: *group* $\mathfrak{H}Pm$ **unfolding** $\mathfrak{H}Pm\text{-def}$ **using** m' $\text{comp}\mathfrak{H}.\text{normal-series-subgroups}$
 $1(2)$ *group.subgroup-imp-group* **by** *force*
have $\text{finGbl:finite}(\text{carrier } \mathfrak{G}Pn)$ **using** $\langle n - 1 < \text{length } \mathfrak{G} \rangle$ $1(3)$ **unfolding**
 $\mathfrak{G}Pn\text{-def}$ **using** $\text{comp}\mathfrak{G}.\text{normal-series-subgroups}$ $\text{comp}\mathfrak{G}.\text{subgroup-finite}$ **by** *auto*
have $\text{finHbl:finite}(\text{carrier } \mathfrak{H}Pm)$ **using** $\langle m - 1 < \text{length } \mathfrak{H} \rangle$ $1(3)$ **unfolding**
 $\mathfrak{H}Pm\text{-def}$ **using** $\text{comp}\mathfrak{H}.\text{normal-series-subgroups}$ $\text{comp}\mathfrak{G}.\text{subgroup-finite}$ **by** *auto*
have $\text{quots}\mathfrak{G}\text{notempty:comp}\mathfrak{G}.\text{quotients} \neq []$ **using** $\text{comp}\mathfrak{G}.\text{quotients-length}$
 length **by** *auto*
have $\text{quots}\mathfrak{H}\text{notempty:comp}\mathfrak{H}.\text{quotients} \neq []$ **using** $\text{comp}\mathfrak{H}.\text{quotients-length}$ $\text{length}\mathfrak{H}\text{big}$
by *auto*
 — Instantiate truncated composition series since they are used for both cases
interpret $\mathfrak{H}\text{butlast}$: *composition-series* $\mathfrak{H}Pm$ **take** m \mathfrak{H} **using** $\text{comp}\mathfrak{H}.\text{composition-series-prefix-closed}$
 $m'(1,2)$ $\mathfrak{H}Pm\text{-def}$ **by** *auto*
interpret $\mathfrak{G}\text{butlast}$: *composition-series* $\mathfrak{G}Pn$ **take** n \mathfrak{G} **using** $\text{comp}\mathfrak{G}.\text{composition-series-prefix-closed}$
 $n'(1,2)$ $\mathfrak{G}Pn\text{-def}$ **by** *auto*
have $\text{ltaken:n} = \text{length}(\text{take } n \ \mathfrak{G})$ **using** $\text{length-take } n'(2)$ **by** *auto*
have $\text{ltakem:m} = \text{length}(\text{take } m \ \mathfrak{H})$ **using** $\text{length-take } m'(2)$ **by** *auto*
show *?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 $\text{lasteq:last comp}\mathfrak{G}.\text{quotients} = \text{last comp}\mathfrak{H}.\text{quotients}$

proof –
from *length* **have** $lg:\text{length } \mathfrak{G} - 1 - 1 + 1 = \text{length } \mathfrak{G} - 1$ **by** (*metis Suc-diff-1 Suc-eq-plus1 n'(1) n-def*)
from *length* \mathfrak{H} *big* **have** $lh:\text{length } \mathfrak{H} - 1 - 1 + 1 = \text{length } \mathfrak{H} - 1$ **by** (*metis Suc-diff-1 Suc-eq-plus1 <0 < m> m-def*)
have *last comp* \mathfrak{G} *.quotients* = $G \text{ Mod } \mathfrak{G} ! (n - 1)$ **using** *length comp* \mathfrak{G} *.last-quotient*
unfolding *n-def* **by** *auto*
also have $\dots = G \text{ Mod } \mathfrak{H} ! (m - 1)$ **using** *True* **by** *simp*
also have $\dots = \text{last comp}$ \mathfrak{H} *.quotients* **using** *length* \mathfrak{H} *big comp* \mathfrak{H} *.last-quotient*
unfolding *m-def* **by** *auto*
finally show *?thesis* .
qed
from *ltaken* **have** $ind:mset (\text{map group.iso-class } \mathfrak{G} \text{butlast.quotients}) = mset (\text{map group.iso-class } \mathfrak{H} \text{butlast.quotients})$
using $1(1) \text{ True } n'(5) \text{ grp}\mathfrak{G}Pn.\text{is-group } \text{finGbl } \mathfrak{G} \text{butlast.is-composition-series } \mathfrak{H} \text{butlast.is-composition-series}$ **unfolding** $\mathfrak{G}Pn\text{-def } \mathfrak{H}Pm\text{-def}$ **by** *metis*
have $mset (\text{map group.iso-class } \text{comp}\mathfrak{G}.\text{quotients}) = mset (\text{map group.iso-class } (\text{butlast } \text{comp}\mathfrak{G}.\text{quotients} @ [\text{last } \text{comp}\mathfrak{G}.\text{quotients}]))$ **by** (*simp add: quots* \mathfrak{G} *notempty*)
also have $\dots = mset (\text{map group.iso-class } (\mathfrak{G} \text{butlast.quotients} @ [\text{last } (\text{comp}\mathfrak{G}.\text{quotients})]))$
using *comp* \mathfrak{G} *.quotients-butlast length* **unfolding** *n-def* $\mathfrak{G}Pn\text{-def}$ **by** *auto*
also have $\dots = mset ((\text{map group.iso-class } \mathfrak{G} \text{butlast.quotients}) @ [\text{group.iso-class } (\text{last } (\text{comp}\mathfrak{G}.\text{quotients}))])$ **by** *auto*
also have $\dots = mset (\text{map group.iso-class } \mathfrak{G} \text{butlast.quotients}) + \{\# \text{ group.iso-class } (\text{last } (\text{comp}\mathfrak{G}.\text{quotients})) \# \}$ **by** *auto*
also have $\dots = mset (\text{map group.iso-class } \mathfrak{H} \text{butlast.quotients}) + \{\# \text{ group.iso-class } (\text{last } (\text{comp}\mathfrak{G}.\text{quotients})) \# \}$ **using** *ind* **by** *simp*
also have $\dots = mset (\text{map group.iso-class } \mathfrak{H} \text{butlast.quotients}) + \{\# \text{ group.iso-class } (\text{last } (\text{comp}\mathfrak{H}.\text{quotients})) \# \}$ **using** *lasteq* **by** *simp*
also have $\dots = mset ((\text{map group.iso-class } \mathfrak{H} \text{butlast.quotients}) @ [\text{group.iso-class } (\text{last } (\text{comp}\mathfrak{H}.\text{quotients}))])$ **by** *auto*
also have $\dots = mset (\text{map group.iso-class } (\mathfrak{H} \text{butlast.quotients} @ [\text{last } (\text{comp}\mathfrak{H}.\text{quotients})]))$
by *auto*
also have $\dots = mset (\text{map group.iso-class } (\text{butlast } \text{comp}\mathfrak{H}.\text{quotients} @ [\text{last } \text{comp}\mathfrak{H}.\text{quotients}]))$ **using** *length* \mathfrak{H} *big comp* \mathfrak{H} *.quotients-butlast* **unfolding** *m-def* $\mathfrak{H}Pm\text{-def}$ **by** *auto*
also have $\dots = mset (\text{map group.iso-class } \text{comp}\mathfrak{H}.\text{quotients})$ **using** *append-butlast-last-id quots* \mathfrak{H} *notempty* **by** *simp*
finally show *?thesis* .
next
case *False*
define $\mathfrak{H}PmInt\mathfrak{G}Pn$ **where** $\mathfrak{H}PmInt\mathfrak{G}Pn = G(\text{carrier} := \mathfrak{H} ! (m - 1) \cap \mathfrak{G} ! (n - 1))$
interpret $\mathfrak{G}Pnmax$: *max-normal-subgroup* $\mathfrak{G} ! (n - 1) G$ **unfolding** *n-def*
by (*metis add-lessD1 diff-diff-add n'(3) add commute one-add-one 1(3) comp* \mathfrak{G} *.snd-to-last-max-normal*)
interpret $\mathfrak{H}Pmmax$: *max-normal-subgroup* $\mathfrak{H} ! (m - 1) G$ **unfolding** *m-def*
by (*metis add-lessD1 diff-diff-add m'(3) add commute one-add-one 1(3) comp* \mathfrak{H} *.snd-to-last-max-normal*)

have $\mathfrak{H}PmnormG:\mathfrak{H}!(m-1) \triangleleft G$ **using** *comp \mathfrak{H} .normal-series-snd-to-last*
m'(4) **unfolding** *m-def* **by** *auto*
have $\mathfrak{G}PnnormG:\mathfrak{G}!(n-1) \triangleleft G$ **using** *comp \mathfrak{G} .normal-series-snd-to-last*
n'(6) **unfolding** *n-def* **by** *auto*
have $\mathfrak{H}Pmint\mathfrak{G}PnnormG:\mathfrak{H}!(m-1) \cap \mathfrak{G}!(n-1) \triangleleft G$ **using** $\mathfrak{H}PmnormG$
 $\mathfrak{G}PnnormG$ **by** (*rule comp \mathfrak{G} .normal-subgroup-intersect*)
have $Intnorm\mathfrak{G}Pn:\mathfrak{H}!(m-1) \cap \mathfrak{G}!(n-1) \triangleleft \mathfrak{G}Pn$ **using** $\mathfrak{G}PnnormG$
 $\mathfrak{H}PmnormG$ *Int-lower2* **unfolding** *\mathfrak{G}Pn-def*
by (*metis comp \mathfrak{G} .normal-restrict-supergroup comp \mathfrak{G} .normal-series-subgroups*
comp \mathfrak{G} .normal-subgroup-intersect n'(4))
then interpret *grp $\mathfrak{G}PnMod\mathfrak{H}Pmint\mathfrak{G}Pn$: group $\mathfrak{G}Pn Mod \mathfrak{H}!(m-1) \cap \mathfrak{G}!$*
(n-1) **by** (*rule normal.factorgroup-is-group*)
have $Intnorm\mathfrak{H}Pm:\mathfrak{H}!(m-1) \cap \mathfrak{G}!(n-1) \triangleleft \mathfrak{H}Pm$ **using** $\mathfrak{H}PmnormG$
 $\mathfrak{G}PnnormG$ *Int-lower2* *Int-commute* **unfolding** *\mathfrak{H}Pm-def*
by (*metis comp \mathfrak{G} .normal-restrict-supergroup comp \mathfrak{G} .normal-subgroup-intersect*
comp \mathfrak{H} .normal-series-subgroups m'(6))
then interpret *grp $\mathfrak{H}PmMod\mathfrak{H}Pmint\mathfrak{G}Pn$: group $\mathfrak{H}Pm Mod \mathfrak{H}!(m-1) \cap$*
 $\mathfrak{G}!(n-1)$ **by** (*rule normal.factorgroup-is-group*)

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

have $not\mathfrak{H}PmSub\mathfrak{G}Pn:\neg(\mathfrak{H}!(m-1) \subseteq \mathfrak{G}!(n-1))$ **using** $\mathfrak{H}Pm-$
max.max-normal $\mathfrak{G}PnnormG$ *False[symmetric]* $\mathfrak{G}Pnmax.proper$ **by** *simp*
have $not\mathfrak{G}PnSub\mathfrak{H}Pm:\neg(\mathfrak{G}!(n-1) \subseteq \mathfrak{H}!(m-1))$ **using** $\mathfrak{G}Pn-$
max.max-normal $\mathfrak{H}PmnormG$ *False* $\mathfrak{H}Pmmax.proper$ **by** *simp*

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

have $\mathfrak{H}PmSubSetmult:\mathfrak{H}!(m-1) \subseteq \mathfrak{H}!(m-1) \langle \# \rangle_G \mathfrak{G}!(n-1)$
using *second-isomorphism-grp.H-contained-in-set-mult* $\mathfrak{G}Pnmax.is-normal$
 $\mathfrak{H}PmnormG$ *normal-imp-subgroup*
unfolding *second-isomorphism-grp-def second-isomorphism-grp-axioms-def*
max-normal-subgroup-def **by** *metis*
have $\mathfrak{G}PnSubSetmult:\mathfrak{G}!(n-1) \subseteq \mathfrak{H}!(m-1) \langle \# \rangle_G \mathfrak{G}!(n-1)$
using *second-isomorphism-grp.S-contained-in-set-mult* $\mathfrak{G}Pnmax.is-normal$
 $\mathfrak{H}PmnormG$ *normal-imp-subgroup*
unfolding *second-isomorphism-grp-def second-isomorphism-grp-axioms-def*
max-normal-subgroup-def **by** *metis*
have $\mathfrak{G}!(n-1) \neq (\mathfrak{H}!(m-1)) \langle \# \rangle_G (\mathfrak{G}!(n-1))$ **using** $\mathfrak{H}PmSubSetmult$
not $\mathfrak{H}PmSub\mathfrak{G}Pn$ **by** *auto*
hence *set-mult G :($\mathfrak{H}!(m-1)) \langle \# \rangle_G (\mathfrak{G}!(n-1)) = carrier G$*
using $\mathfrak{G}Pnmax.max-normal$ $\mathfrak{G}Pnmax.is-normal$ $\mathfrak{H}PmnormG$ *comp \mathfrak{G} .normal-subgroup-set-mult-closed*
 $\mathfrak{G}PnSubSetmult$ **by** *metis*
then obtain φ **where** $\varphi \in iso(\mathfrak{G}Pn Mod (\mathfrak{H}!(m-1) \cap \mathfrak{G}!(n-1)))$
 $(G \setminus \{carrier\} := carrier G) Mod \mathfrak{H}!(m-1)$
using *second-isomorphism-grp.normal-intersection-quotient-isom* $\mathfrak{H}Pm-$
normG $\mathfrak{G}Pnmax.is-normal$ *normal-imp-subgroup*
unfolding *second-isomorphism-grp-def second-isomorphism-grp-axioms-def*
max-normal-subgroup-def *\mathfrak{G}Pn-def* **by** *metis*
hence $\varphi:\varphi \in iso(\mathfrak{G}Pn Mod (\mathfrak{H}!(m-1) \cap \mathfrak{G}!(n-1))) (G Mod \mathfrak{H}!(m-$
 $-1))$ **by** *auto*

then obtain φ_2 **where** $\varphi_2:\varphi_2 \in iso (G \text{ Mod } \mathfrak{H}! (m - 1)) (\mathfrak{G}Pn \text{ Mod } (\mathfrak{H}! (m - 1) \cap \mathfrak{G}! (n - 1)))$
using *group.iso-set-sym grp $\mathfrak{G}Pn\text{Mod}\mathfrak{H}Pmint\mathfrak{G}Pn.is-group$ by auto*
moreover have *simple-group (G($\backslash carrier := \mathfrak{H}! (m - 1 + 1)$) Mod $\mathfrak{H}! (m - 1)$) using comp $\mathfrak{H}.simplefact m'(3)$ by simp*
hence *simple-group (G Mod $\mathfrak{H}! (m - 1)$) using comp $\mathfrak{H}.last last-conv-nth comp\mathfrak{H}.notempty m'(5)$ by fastforce*
ultimately have *simple $\mathfrak{G}Pn\text{ModInt}:simple-group (\mathfrak{G}Pn \text{ Mod } (\mathfrak{H}! (m - 1) \cap \mathfrak{G}! (n - 1)))$*
using *simple-group.iso-simple grp $\mathfrak{G}Pn\text{Mod}\mathfrak{H}Pmint\mathfrak{G}Pn.is-group$ by auto*
interpret *grpGMod $\mathfrak{H}Pm$: group (G Mod $\mathfrak{H}! (m - 1)$) by (metis $\mathfrak{H}PmnormG normal.factorgroup-is-group$)*

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

have $\mathfrak{H}PmSubSetmult':\mathfrak{H}! (m - 1) \subseteq \mathfrak{G}! (n - 1) \langle \# \rangle_G \mathfrak{H}! (m - 1)$
using *second-isomorphism-grp.S-contained-in-set-mult $\mathfrak{G}Pnmax.is-normal \mathfrak{H}PmnormG normal-imp-subgroup$*
unfolding *second-isomorphism-grp-def second-isomorphism-grp-axioms-def max-normal-subgroup-def by metis*
have $\mathfrak{G}PnSubSetmult':\mathfrak{G}! (n - 1) \subseteq \mathfrak{G}! (n - 1) \langle \# \rangle_G \mathfrak{H}! (m - 1)$
using *second-isomorphism-grp.H-contained-in-set-mult $\mathfrak{G}Pnmax.is-normal \mathfrak{H}PmnormG normal-imp-subgroup$*
unfolding *second-isomorphism-grp-def second-isomorphism-grp-axioms-def max-normal-subgroup-def by metis*
have $\mathfrak{H}! (m - 1) \neq (\mathfrak{G}! (n - 1)) \langle \# \rangle_G (\mathfrak{H}! (m - 1))$ **using** *$\mathfrak{G}PnSubSetmult'$ not $\mathfrak{G}PnSub\mathfrak{H}Pm$ by auto*
hence *set-multG:($\mathfrak{G}! (n - 1)$) $\langle \# \rangle_G (\mathfrak{H}! (m - 1)) = carrier G$*
using *$\mathfrak{H}Pmmax.max-normal \mathfrak{H}Pmmax.is-normal \mathfrak{G}PnnormG comp\mathfrak{G}.normal-subgroup-set-mult-closed \mathfrak{H}PmSubSetmult'$ by metis*
from *set-multG* **obtain** ψ **where**
 $\psi \in iso (\mathfrak{H}Pm \text{ Mod } (\mathfrak{G}! (n - 1) \cap \mathfrak{H}! (m - 1))) (G(\backslash carrier := carrier G) \text{ Mod } \mathfrak{G}! (n - 1))$
using *second-isomorphism-grp.normal-intersection-quotient-isom $\mathfrak{G}PnnormG \mathfrak{H}Pmmax.is-normal normal-imp-subgroup$*
unfolding *second-isomorphism-grp-def second-isomorphism-grp-axioms-def max-normal-subgroup-def $\mathfrak{H}Pm-def$ by metis*
hence $\psi:\psi \in iso (\mathfrak{H}Pm \text{ Mod } (\mathfrak{H}! (m - 1) \cap (\mathfrak{G}! (n - 1)))) (G(\backslash carrier := carrier G) \text{ Mod } \mathfrak{G}! (n - 1))$ **using** *Int-commute by metis*
then obtain ψ_2 **where**
 $\psi_2:\psi_2 \in iso (G \text{ Mod } \mathfrak{G}! (n - 1)) (\mathfrak{H}Pm \text{ Mod } (\mathfrak{H}! (m - 1) \cap \mathfrak{G}! (n - 1)))$
using *group.iso-set-sym grp $\mathfrak{H}Pm\text{Mod}\mathfrak{H}Pmint\mathfrak{G}Pn.is-group$ by auto*
moreover have *simple-group (G($\backslash carrier := \mathfrak{G}! (n - 1 + 1)$) Mod $\mathfrak{G}! (n - 1)$) using comp $\mathfrak{G}.simplefact n'(3)$ by simp*
hence *simple-group (G Mod $\mathfrak{G}! (n - 1)$) using comp $\mathfrak{G}.last last-conv-nth comp\mathfrak{G}.notempty n'(7)$ by fastforce*
ultimately have *simple $\mathfrak{H}Pm\text{ModInt}:simple-group (\mathfrak{H}Pm \text{ Mod } (\mathfrak{H}! (m - 1) \cap \mathfrak{G}! (n - 1)))$*

using *simple-group.iso-simple grp \mathfrak{H} PmMod \mathfrak{H} Pmint \mathfrak{G} Pn.is-group* **by** *auto*
interpret *grpGMod \mathfrak{G} Pn: group (G Mod $\mathfrak{G} ! (n - 1)$)* **by** (*metis \mathfrak{G} PnnormG*
normal.factorgroup-is-group)

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

define \mathfrak{K} **where** $\mathfrak{K} = \text{remdups-adj } (\text{map } ((\cap) (\mathfrak{H} ! (m - 1))) \mathfrak{G})$
define \mathfrak{L} **where** $\mathfrak{L} = \text{remdups-adj } (\text{map } ((\cap) (\mathfrak{G} ! (n - 1))) \mathfrak{H})$
interpret \mathfrak{K} : *composition-series \mathfrak{H} Pm* \mathfrak{K} **using** *comp \mathfrak{G} .intersect-normal 1(3)*
 \mathfrak{H} PnnormG **unfolding** \mathfrak{K} -def \mathfrak{H} Pm-def **by** *auto*
interpret \mathfrak{L} : *composition-series \mathfrak{G} Pn* \mathfrak{L} **using** *comp \mathfrak{H} .intersect-normal 1(3)*
 \mathfrak{G} PnnormG **unfolding** \mathfrak{L} -def \mathfrak{G} Pn-def **by** *auto*

— Apply the induction hypothesis on \mathfrak{G} butlast and \mathfrak{L}

from $n'(2)$ **have** *Suc (length (take n \mathfrak{G})) \leq length \mathfrak{G}* **by** *auto*
hence *multisets \mathfrak{G} butlast \mathfrak{L} :mset (map group.iso-class \mathfrak{G} butlast.quotients) =*
mset (map group.iso-class \mathfrak{L} .quotients)
using *1.hyps grp \mathfrak{G} Pn.is-group finGbl \mathfrak{G} butlast.is-composition-series \mathfrak{L} .is-composition-series*
by *metis*
hence *length \mathfrak{L} :n = length \mathfrak{L}* **using** *\mathfrak{G} butlast.quotients-length \mathfrak{L} .quotients-length*
length-map size-mset ltaken **by** *metis*
hence *length \mathfrak{L}' :length $\mathfrak{L} > 1$ length $\mathfrak{L} - 1 > 0$ length $\mathfrak{L} - 1 \leq$ length \mathfrak{L}*
using $n'(6)$ *length* **by** *auto*
have *Inteq \mathfrak{L} sndlast: $\mathfrak{H} ! (m - 1) \cap \mathfrak{G} ! (n - 1) = \mathfrak{L} ! (\text{length } \mathfrak{L} - 1 - 1)$*
proof —
have *length $\mathfrak{L} - 1 - 1 + 1 <$ length \mathfrak{L}* **using** *length \mathfrak{L}'* **by** *auto*
moreover **have** *KGnotempty:(map ((\cap) ($\mathfrak{G} ! (n - 1)$)) $\mathfrak{H}) \neq \square$* **using**
comp \mathfrak{H} .notempty **by** (*metis Nil-is-map-conv*)
ultimately obtain i **where** *$i : i + 1 <$ length (map ((\cap) ($\mathfrak{G} ! (n - 1)$)) $\mathfrak{H})$*
 $\mathfrak{L} ! (\text{length } \mathfrak{L} - 1 - 1) = (\text{map } ((\cap) (\mathfrak{G} ! (n - 1))) \mathfrak{H}) ! i \mathfrak{L} ! (\text{length } \mathfrak{L}$
 $- 1 - 1 + 1) = (\text{map } ((\cap) (\mathfrak{G} ! (n - 1))) \mathfrak{H}) ! (i + 1)$
using *remdups-adj-obtain-adjacency* **unfolding** \mathfrak{L} -def **by** *force*
hence *$\mathfrak{L} ! (\text{length } \mathfrak{L} - 1 - 1) = \mathfrak{H} ! i \cap \mathfrak{G} ! (n - 1) \mathfrak{L} ! (\text{length } \mathfrak{L} - 1 -$*
 $1 + 1) = \mathfrak{H} ! (i + 1) \cap \mathfrak{G} ! (n - 1)$ **by** *auto*
hence *$\mathfrak{L} ! (\text{length } \mathfrak{L} - 1) = \mathfrak{H} ! (i + 1) \cap \mathfrak{G} ! (n - 1)$* **using** *length $\mathfrak{L}'(2)$*
by (*metis Suc-diff-1 Suc-eq-plus1*)
hence *\mathfrak{G} Pnsub \mathfrak{H} Pm: $\mathfrak{G} ! (n - 1) \subseteq \mathfrak{H} ! (i + 1)$* **using** *$\mathfrak{L}$.last \mathfrak{L} .notempty*
last-conv-nth **unfolding** \mathfrak{G} Pn-def **by** *auto*
from $i(1)$ **have** *$i + 1 <$ $m + 1$* **unfolding** m -def **by** *auto*
moreover **have** *$\neg (i + 1 \leq m - 1)$* **using** *comp \mathfrak{H} .entries-mono $m'(6)$*
not \mathfrak{G} PnSub \mathfrak{H} Pm \mathfrak{G} Pnsub \mathfrak{H} Pm **by** *fastforce*
ultimately have *$m - 1 = i$* **by** *auto*
with i **show** *?thesis* **by** *auto*
qed
hence *\mathfrak{L} sndlast: \mathfrak{H} PmInt \mathfrak{G} Pn = (\mathfrak{G} Pn($\text{carrier} := \mathfrak{L} ! (\text{length } \mathfrak{L} - 1 - 1)$))*
unfolding \mathfrak{H} PmInt \mathfrak{G} Pn-def \mathfrak{G} Pn-def **by** *auto*
then interpret \mathfrak{L} butlast: *composition-series \mathfrak{H} PmInt \mathfrak{G} Pn* **take** (*length $\mathfrak{L} -$*
1) \mathfrak{L} **using** *length \mathfrak{L}' \mathfrak{L} .composition-series-prefix-closed* **by** *metis*

from $\langle \text{length } \mathcal{L} > 1 \rangle$ **have** $\text{quots } \mathcal{L} \text{ not empty} : \mathcal{L}.\text{quotients} \neq []$ **unfolding**
 $\mathcal{L}.\text{quotients-def}$ **by** *auto*

— Apply the induction hypothesis on $\mathcal{L}\text{butlast}$ and $\mathcal{R}\text{butlast}$

have $\text{length } \mathcal{R} > 1$
proof (*rule ccontr*)
assume $\neg \text{length } \mathcal{R} > 1$
with $\mathcal{R}.\text{notempty}$ **have** $\text{length } \mathcal{R} = 1$ **by** (*metis One-nat-def Suc-less1*
 $\text{length-greater-0-conv}$)
hence $\text{carrier } \mathfrak{H}Pm = \{1_{\mathfrak{H}Pm}\}$ **using** $\mathcal{R}.\text{composition-series-length-one}$
 $\mathcal{R}.\text{composition-series-triv-group}$ **by** *auto*
hence $\text{carrier } \mathfrak{H}Pm = \{1_G\}$ **unfolding** $\mathfrak{H}Pm\text{-def}$ **by** *auto*
hence $\text{carrier } \mathfrak{H}Pm \subseteq \mathfrak{G} ! (n - 1)$ **using** $\mathfrak{G}Pn.\text{max.is-subgroup sub-}$
 group.one-closed **by** *auto*
with $\text{not } \mathfrak{H}Pm\text{Sub } \mathfrak{G}Pn$ **show** *False* **unfolding** $\mathfrak{H}Pm\text{-def}$ **by** *auto*
qed
hence $\text{length } \mathcal{R}' : \text{length } \mathcal{R} - 1 > 0$ $\text{length } \mathcal{R} - 1 \leq \text{length } \mathcal{R}$ **by** *auto*
have $\text{Inteq } \mathcal{R}.\text{sndlast} : \mathfrak{H} ! (m - 1) \cap \mathfrak{G} ! (n - 1) = \mathcal{R} ! (\text{length } \mathcal{R} - 1 - 1)$
proof —
have $\text{length } \mathcal{R} - 1 - 1 + 1 < \text{length } \mathcal{R}$ **using** $\text{length } \mathcal{R}'$ **by** *auto*
moreover **have** $\text{KGnotempty} : (\text{map } ((\cap) (\mathfrak{H} ! (m - 1))) \mathfrak{G}) \neq []$ **using**
 $\text{comp } \mathfrak{G}.\text{notempty}$ **by** (*metis Nil-is-map-conv*)
ultimately obtain i **where** $i : i + 1 < \text{length } (\text{map } ((\cap) (\mathfrak{H} ! (m - 1))) \mathfrak{G})$
 $\mathcal{R} ! (\text{length } \mathcal{R} - 1 - 1) = (\text{map } ((\cap) (\mathfrak{H} ! (m - 1))) \mathfrak{G}) ! i$ $\mathcal{R} ! (\text{length } \mathcal{R} - 1 - 1 + 1) = (\text{map } ((\cap) (\mathfrak{H} ! (m - 1))) \mathfrak{G}) ! (i + 1)$
using $\text{remdups-adj-obtain-adjacency}$ **unfolding** $\mathcal{R}\text{-def}$ **by** *force*
hence $\mathcal{R} ! (\text{length } \mathcal{R} - 1 - 1) = \mathfrak{G} ! i \cap \mathfrak{H} ! (m - 1)$ $\mathcal{R} ! (\text{length } \mathcal{R} - 1 - 1 + 1) = \mathfrak{G} ! (i + 1) \cap \mathfrak{H} ! (m - 1)$ **by** *auto*
hence $\mathcal{R} ! (\text{length } \mathcal{R} - 1) = \mathfrak{G} ! (i + 1) \cap \mathfrak{H} ! (m - 1)$ **using** $\text{length } \mathcal{R}'(1)$
by (*metis Suc-diff-1 Suc-eq-plus1*)
hence $\mathfrak{H}Pm\text{sub } \mathfrak{G}Pn : \mathfrak{H} ! (m - 1) \subseteq \mathfrak{G} ! (i + 1)$ **using** $\mathcal{R}.\text{last}$ $\mathcal{R}.\text{notempty}$
 last-conv-nth **unfolding** $\mathfrak{H}Pm\text{-def}$ **by** *auto*
from $i(1)$ **have** $i + 1 < n + 1$ **unfolding** $n\text{-def}$ **by** *auto*
moreover **have** $\neg (i + 1 \leq n - 1)$ **using** $\text{comp } \mathfrak{G}.\text{entries-mono } n'(2)$
 $\text{not } \mathfrak{H}Pm\text{Sub } \mathfrak{G}Pn$ $\mathfrak{H}Pm\text{sub } \mathfrak{G}Pn$ **by** *fastforce*
ultimately **have** $n - 1 = i$ **by** *auto*
with i **show** *?thesis* **by** *auto*
qed
have $\text{composition-series } (G \setminus \text{carrier} := \mathcal{R} ! (\text{length } \mathcal{R} - 1 - 1))$ ($\text{take } (\text{length } \mathcal{R} - 1) \mathcal{R}$)
using $\text{length } \mathcal{R}'$ $\mathcal{R}.\text{composition-series-prefix-closed}$ **unfolding** $\mathfrak{H}Pm\text{Int } \mathfrak{G}Pn\text{-def}$
 $\mathfrak{H}Pm\text{-def}$ **by** *fastforce*
then interpret $\mathcal{R}\text{butlast} : \text{composition-series } \mathfrak{H}Pm\text{Int } \mathfrak{G}Pn$ ($\text{take } (\text{length } \mathcal{R} - 1) \mathcal{R}$) **using** $\text{Inteq } \mathcal{R}.\text{sndlast}$ **unfolding** $\mathfrak{H}Pm\text{Int } \mathfrak{G}Pn\text{-def}$ **by** *auto*
from finGbl **have** $\text{finInt} : \text{finite } (\text{carrier } \mathfrak{H}Pm\text{Int } \mathfrak{G}Pn)$ **unfolding** $\mathfrak{H}Pm\text{Int } \mathfrak{G}Pn\text{-def}$
 $\mathfrak{G}Pn\text{-def}$ **by** *simp*
moreover **have** $\text{Suc } (\text{length } (\text{take } (\text{length } \mathcal{L} - 1) \mathcal{L})) \leq \text{length } \mathfrak{G}$ **using**
 $\text{length } \mathcal{L}$ **unfolding** $n\text{-def}$ **using** $n'(2)$ **by** *auto*
ultimately **have** $\text{multisets } \mathcal{R}\mathcal{L}\text{butlast} : \text{mset } (\text{map } \text{group.iso-class } \mathcal{L}\text{butlast.quotients})$

= *mset* (*map group.iso-class* *Rbutlast.quotients*)
 using *1.hyps* *Lbutlast.is-group Rbutlast.is-composition-series Lbutlast.is-composition-series*
by *auto*
 hence *length* (*take* (*length* *R* - 1) *R*) = *length* (*take* (*length* *L* - 1) *L*)
 using *Rbutlast.quotients-length Lbutlast.quotients-length length-map size-mset*
by *metis*
 hence *length* (*take* (*length* *R* - 1) *R*) = *n* - 1 **using** *lengthL n'(1)* **by** *auto*
 hence *lengthR: length R = n* **by** (*metis Suc-diff-1 R.notempty butlast-conv-take length-butlast length-greater-0-conv n'(1)*)

— Apply the induction hypothesis on *R* and *Hbutlast*
 from *InteqRsndlast* **have** *Rsndlast: HPMIntGPn* = (*HPM* (*carrier := R !*
 (*length R - 1 - 1*))) **unfolding** *HPMIntGPn-def HPM-def R-def* **by** *auto*
 from *lengthR* **have** *Suc* (*length R*) ≤ *length G* **using** *n'(2)* **by** *auto*
 hence *multisetsHbutlastR: mset* (*map group.iso-class Hbutlast.quotients*) =
mset (*map group.iso-class R.quotients*)
 using *1.hyps grpHPM.is-group finHbl Hbutlast.is-composition-series R.is-composition-series*
by *metis*
 hence *lengthR:m = length R* **using** *Hbutlast.quotients-length R.quotients-length length-map size-mset ltakem* **by** *metis*
 hence *length R > 1* *length R - 1 > 0* *length R - 1 ≤ length R* **using** *m'(4)*
lengthHbig **by** *auto*
 hence *quotsRnotempty: R.quotients ≠ []* **unfolding** *R.quotients-def* **by** *auto*

interpret *RbutlastaddGPn: composition-series GPn* (*take* (*length R - 1*) *R*)
 @ [*G ! (n - 1)*]
 using *grpGPn.composition-series-extend Rbutlast.is-composition-series simpleGPnModInt IntnormGPn*
 unfolding *GPn-def HPMIntGPn-def* **by** *auto*
 interpret *LbutlastaddHPM: composition-series HPM* (*take* (*length L - 1*) *L*)
 @ [*H ! (m - 1)*]
 using *grpHPM.composition-series-extend Lbutlast.is-composition-series simpleHPMModInt IntnormHPM*
 unfolding *HPM-def HPMIntGPn-def* **by** *auto*

— Prove equality of those composition series.
 have *mset* (*map group.iso-class compG.quotients*)
 = *mset* (*map group.iso-class* ((*butlast compG.quotients*) @ [*last*
compG.quotients])) **using** *quotsGnotempty* **by** *simp*
 also have ... = *mset* (*map group.iso-class* (*Gbutlast.quotients* @ [*G Mod G*
! (n - 1)]))
 using *compG.quotients-butlast compG.last-quotient length* **unfolding** *n-def GPn-def* **by** *auto*
 also have ... = *mset* (*map group.iso-class* ((*butlast L.quotients*) @ [*last*
L.quotients])) + {*# group.iso-class* (*G Mod G ! (n - 1)*) #}
 using *multisetsGbutlastL quotsLnotempty* **by** *simp*
 also have ... = *mset* (*map group.iso-class* (*Lbutlast.quotients* @ [*GPn Mod*
H ! (m - 1) ∩ G ! (n - 1)])) + {*# group.iso-class* (*G Mod G ! (n - 1)*) #}
 using *L.quotients-butlast L.last-quotient <length L > 1>* *Lsndlast In-*

$teq\mathcal{L}sndlast$ **unfolding** n -def by auto
also have $\dots = mset (map\ group.iso-class\ \mathcal{R}butlast.quotients) + \{\# group.iso-class\ (\mathcal{G}Pn\ Mod\ \mathfrak{H}!\ (m-1) \cap \mathcal{G}!\ (n-1))\ \#\} + \{\# group.iso-class\ (G\ Mod\ \mathcal{G}!\ (n-1))\ \#\}$
using $multisets\mathcal{R}\mathcal{L}butlast$ by simp
also have $\dots = mset (map\ group.iso-class\ \mathcal{R}butlast.quotients) + \{\# group.iso-class\ (G\ Mod\ \mathfrak{H}!\ (m-1))\ \#\} + \{\# group.iso-class\ (\mathfrak{H}Pm\ Mod\ \mathfrak{H}!\ (m-1) \cap \mathcal{G}!\ (n-1))\ \#\}$
using $\varphi\ \psi 2\ iso-classes-iff\ grp\mathcal{G}PnMod\mathfrak{H}Pmint\mathcal{G}Pn.is-group\ grpGMod\mathfrak{H}Pm.is-group\ grpGMod\mathcal{G}Pn.is-group\ grp\mathfrak{H}PmMod\mathfrak{H}Pmint\mathcal{G}Pn.is-group$
by metis
also have $\dots = mset (map\ group.iso-class\ \mathcal{R}butlast.quotients) + \{\# group.iso-class\ (\mathfrak{H}Pm\ Mod\ \mathfrak{H}!\ (m-1) \cap \mathcal{G}!\ (n-1))\ \#\} + \{\# group.iso-class\ (G\ Mod\ \mathfrak{H}!\ (m-1))\ \#\}$
by simp
also have $\dots = mset (map\ group.iso-class\ ((butlast\ \mathcal{R}.quotients) @ [last\ \mathcal{R}.quotients])) + \{\# group.iso-class\ (G\ Mod\ \mathfrak{H}!\ (m-1))\ \#\}$
using $\mathcal{R}.quotients-butlast\ \mathcal{R}.last-quotient\ \langle length\ \mathcal{R} > 1 \rangle\ \mathcal{R}sndlast\ Inteq\mathcal{R}sndlast$ **unfolding** m -def by auto
also have $\dots = mset (map\ group.iso-class\ \mathfrak{H}butlast.quotients) + \{\# group.iso-class\ (G\ Mod\ \mathfrak{H}!\ (m-1))\ \#\}$
using $multisets\mathfrak{H}butlast\mathcal{R}\ quotes\mathcal{R}notempty$ by simp
also have $\dots = mset (map\ group.iso-class\ ((butlast\ comp\mathfrak{H}.quotients) @ [last\ comp\mathfrak{H}.quotients]))$
using $comp\mathfrak{H}.quotients-butlast\ comp\mathfrak{H}.last-quotient\ length\mathfrak{H}big$ **unfolding** m -def $\mathfrak{H}Pm$ -def by auto
also have $\dots = mset (map\ group.iso-class\ comp\mathfrak{H}.quotients)$ **using** $quotes\mathfrak{H}notempty$
by simp
finally show $?thesis$.
qed
qed
qed

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

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

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

also have $\dots = size\ (mset\ (map\ group.iso-class\ comp\mathfrak{H}.quotients)) + 1$

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

also have $\dots = length\ (map\ group.iso-class\ comp\mathfrak{H}.quotients) + 1$ by $(metis\ size-mset)$

also have $\dots = length\ comp\mathfrak{H}.quotients + 1$ by $(metis\ length-map)$

also have $\dots = \text{length } \mathfrak{H}$ **by** (*metis comp* \mathfrak{H} .*quotients-length*)
finally show *?thesis*.
qed
end

References

- [Ran05] Stuart Rankin. The jordan-hölder theorem, 2005.
- [vR14] Jakob von Raumer. Secondary sylow theorems. *Archive of Formal Proofs*, January 2014. http://isa-afp.org/entries/Secondary_Sylow.shtml, Formal proof development.