The Jordan-Hölder Theorem

Jakob von Raumer

February 23, 2021

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 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

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* (λ*x*. *x*) **unfolding** *group-hom-def group-hom-axioms-def hom-def* **using** *is-group* **by** *simp*

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

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

lemma (**in** *group*) *self-factor-iso*:
shows ($\lambda X.$ *the-elem* (($\lambda x.$ **1**) ‘ X)) ∈ *iso* (G *Mod* (*carrier* G)) (G⦇ *carrier* := {**1**} ⦈)
proof −
have *group* (G⦇*carrier* := {**1**}⦈) **by** (*metis subgroup-imp-group triv-subgroup*)
hence *group-hom* G (G⦇*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**) ‘ *carrier* G = *carrier* (G⦇*carrier* := {**1**}⦈) **by** *auto*
moreover have *kernel* G (G⦇*carrier* := {**1**}⦈) ($\lambda x.$ **1**) = *carrier* G **unfolding** *kernel-def* **by** *auto*
ultimately show *?thesis* **using** *group-hom.FactGroup-iso-set* **by** *force*
qed

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

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

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

context *second-isomorphism-grp*
begin

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

lemma *normal-subgrp-intersection-normal*:
shows $S \cap H \lhd$ (G⦇*carrier* := S⦈)
proof(*auto simp*: *groupS.normal-inv-iff*)
from *subgrpS is-subgroup* **have** $\bigwedge x.\ x \in \{S, H\} \Longrightarrow$ *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⦇*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 *subgrpS* **show** $g \otimes h \otimes inv_{G⦇carrier := S⦈}\ g \in H$ **by** (*metis inv-op-closed2 subgroup.mem-carrier m-inv-consistent*)

} {
from *g hS subgrpS* **show** $g \otimes h \otimes inv_{G(\!|carrier := S|\!)}\ g \in S$ **by** (*metis subgroup.m-closed subgroup.m-inv-closed m-inv-consistent*)
}
qed

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

lemma *oneH*:$\mathbf{1} \in H$ **by** (*metis is-subgroup subgroup.one-closed*)

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

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

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

lemma *normal-intersection-hom-kernel*:
shows *kernel* (G(|*carrier* := S|)) (((G(|*carrier* := $H <\#> S$|)) *Mod* H) (λg. $H \#> g$) = $H \cap S$
proof −

have *kernel* $(G(\!|carrier := S|\!))$ $((G(\!|carrier := H <\#> S|\!))$ *Mod* $H)$ $(\lambda g.\ H\ \#>\ g)$
$= \{g \in S.\ H\ \#>\ g = \mathbf{1}_{(G(\!|carrier := H <\#> S|\!))\ Mod\ H}\}$ **unfolding** *kernel-def* **by** *auto*
also have $... = \{g \in S.\ H\ \#>\ g = H\}$ **unfolding** *FactGroup-def* **by** *auto*
also have $... = \{g \in S.\ g \in H\}$ **by** (*metis coset-eq is-subgroup lcoset-join2 rcos-self subgroup.mem-carrier subgrpS*)
also have $... = H \cap S$ **by** *auto*
finally show *?thesis*.
qed

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

Finally we can prove the actual isomorphism theorem:

theorem *normal-intersection-quotient-isom*:
shows $(\lambda X.\ the\text{-}elem\ ((\lambda g.\ H\ \#>\ g)$ ' $X)) \in iso\ ((G(\!|carrier := S|\!))\ Mod\ (H \cap S))\ (((G(\!|carrier := H <\#> S|\!)))\ Mod\ H)$
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 (*order G = 1*) = (*carrier G* = {**1**})
proof
assume *order*:*order G = 1*
then obtain *x* **where** *x*:*carrier G* = {*x*} **unfolding** *order-def* **by** (*auto simp add*: *card-Suc-eq*)
hence 1 = *x* **using** *one-closed* **by** *auto*
with *x* **show** *carrier G* = {**1**} **by** *simp*
next
assume *carrier G* = {**1**}
thus *order G = 1* **unfolding** *order-def* **by** *auto*
qed

lemma (**in** *group*) *finite-pos-order*:
assumes *finite*:*finite* (*carrier G*)
shows *0 < 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 *φ* ∈ *iso G H*
shows *order G = 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*(| *carrier* := *H* |))
shows *U* ⊆ *H*
using *assms subgroup.subset* **by** *force*

lemma (**in** *subgroup*) *subgroup-of-subgroup*:
assumes *group G*
assumes *subgroup U* (*G*(| *carrier* := *H* |))
shows *subgroup U G*
proof
from *assms*(*2*) **have** *U* ⊆ *H* **by** (*rule subgroup-of-restricted-group*)
thus *U* ⊆ *carrier G* **by** (*auto simp*:*subset*)
next

fix x y
have a:$x \otimes y = x \otimes_{G(\!|\ carrier := H|\!)}\ y$ **by** *simp*
assume $x \in U$ $y \in U$
with *assms a* **show** $x \otimes y \in U$ **by** (*metis subgroup.m-closed*)
next
have $\mathbf{1}_{G(\!|\ carrier := H|\!)} = \mathbf{1}$ **by** *simp*
with *assms* **show** $\mathbf{1} \in U$ **by** (*metis subgroup.one-closed*)
next
have *subgroup H G*..
fix x
assume $x \in U$
with *assms*(*2*) **have** $inv_{G(\!|\ carrier := H|\!)}\ x \in U$ **by** (*rule subgroup.m-inv-closed*)
moreover from *assms* ‹$x \in U$› **have** $x \in H$ **by** (*metis in-mono subgroup-of-restricted-group*)
with *assms*(*1*) ‹*subgroup H G*› **have** $inv_{G(\!|\ carrier := H|\!)}\ x = inv\ x$ **by** (*rule group.m-inv-consistent*)
ultimately show $inv\ x \in U$ **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 φ*surj*:φ ‘ (*carrier G*) = *carrier F*
shows *subgroup* (φ ‘ *H*) *F*
proof
from φ*surj* **show** *img-subset*:φ ‘ *H* ⊆ *carrier F* **unfolding** *iso-def bij-betw-def*
by *auto*
next
fix *f f'*
assume *h*:$f \in \varphi$ ‘ *H* **and** *h'*:$f' \in \varphi$ ‘ *H*
with φ*surj* **obtain** *g g'* **where** *g*:$g \in H\ f = \varphi\ g$ **and** *g'*:$g' \in H\ f' = \varphi\ g'$ **by** *auto*
hence $g \otimes_G g' \in H$ **by** (*metis m-closed*)
hence $\varphi\ (g \otimes_G g') \in \varphi$ ‘ *H* **by** *simp*
with *g g'* φ **show** $f \otimes_F f' \in \varphi$ ‘ *H* **using** *group-hom.hom-mult* **by** *fastforce*
next
have $\varphi\ \mathbf{1} \in \varphi$ ‘ *H* **by** *auto*
with φ **show** $\mathbf{1}_F \in \varphi$ ‘ *H* **by** (*metis group-hom.hom-one*)
next
fix *f*
assume *f*:$f \in \varphi$ ‘ *H*
then obtain *g* **where** *g*:$g \in H\ f = \varphi\ g$ **by** *auto*
hence *inv g* ∈ *H* **by** *auto*
hence φ (*inv g*) ∈ φ ‘ *H* **by** *auto*
with φ *g subset* **show** $inv_F\ f \in \varphi$ ‘ *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*:*subgroup H G*
assumes φ:$\varphi \in$ *iso G F*
shows *subgroup* (φ ‘ *H*) *F*
proof −
from *groups* φ **have** *group-hom G F* φ **unfolding** *group-hom-def group-hom-axioms-def iso-def* **by** *auto*
moreover from φ **have** φ ‘ (*carrier G*) = *carrier F* **unfolding** *iso-def bij-betw-def* **by** *simp*
moreover note *HG*
ultimately show *?thesis* **by** (*metis subgroup.surj-hom-subgroup*)
qed

An isomorphism restricts to an isomorphism of subgroups.

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

The intersection of two subgroups is, again, a subgroup

lemma (**in** *group*) *subgroup-intersect*:
assumes *subgroup H G*
assumes *subgroup H′ G*
shows *subgroup* (*H* ∩ *H′*) *G*
using *assms* **unfolding** *subgroup-def* **by** *auto*

4 Facts about Normal Subgroups

lemma (**in** *normal*) *is-normal*:
shows *H* ◁ *G*
by (*metis coset-eq is-subgroup 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 (φ ‘ *H*) $\lhd$ *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* (φ ‘ *H*) *F* **by** (*rule surj-hom-subgroup*)
next
show $\forall x \in$*carrier F*. φ ‘ *H* #>$_F$ *x* = *x* <#$_F$ φ ‘ *H*
proof
fix *f*
assume *f*:*f* $\in$ *carrier F*
with φ*surj* **obtain** *g* **where** *g*:*g* $\in$ *carrier G f* = φ *g* **by** *auto*
hence φ ‘ *H* #>$_F$ *f* = φ ‘ *H* #>$_F$ φ *g* **by** *simp*
also have ... = (λx. (φ *x*) $\otimes_F$ (φ *g*)) ‘ *H* **unfolding** *r-coset-def image-def* **by** *auto*
also have ... = (λx. φ (*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 ... = (λx. φ (*g* $\otimes$ *x*)) ‘ *H* **using** φ **unfolding** *l-coset-def* **by** *auto*
also have ... = (λx. (φ *g*) $\otimes_F$ (φ *x*)) ‘ *H* **using** *subset g* φ *group-hom.hom-mult* **by** *fastforce*
also have ... = φ *g* <#$_F$ φ ‘ *H* **unfolding** *l-coset-def image-def* **by** *auto*
also have ... = *f* <#$_F$ φ ‘ *H* **using** *g* **by** *simp*
finally show φ ‘ *H* #>$_F$ *f* = *f* <#$_F$ φ ‘ *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* $\lhd$ *G*
assumes φ:φ $\in$ *iso G F*
shows (φ ‘ *H*) $\lhd$ *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 *card* (*rcosets* {**1**}) = *order G*
proof −
have *subgroup* {**1**} *G* **by** (*rule triv-subgroup*)
with *assms* **have** *card* (*rcosets* {**1**}) ∗ *card* {**1**} = *order G*
using *lagrange* **by** *blast*
thus *?thesis* **by** (*auto simp*:*card-Suc-eq*)
qed

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

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

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

lemma (**in** *group*) *setmult-lcos-assoc*:
⟦*H* ⊆ *carrier G*; *K* ⊆ *carrier G*; *x* ∈ *carrier G*⟧
⟹ (*x* <# *H*) <#> *K* = *x* <# (*H* <#> *K*)
by (*force simp add*: *l-coset-def set-mult-def m-assoc*)

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

qed

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

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

A subgroup relation survives factoring by a normal subgroup.

lemma (**in** *group*) *normal-subgroup-factorize*:
assumes $N \lhd G$ **and** $N \subseteq H$ **and** *subgroup H G*
shows *subgroup* ($rcosets_{G(\!|carrier := H|\!)}\ N$) (*G Mod N*)
proof −
interpret *GModN*: *group G Mod N* **using** *assms(1)* **by** (*rule normal.factorgroup-is-group*)
have $N \lhd G(\!|$*carrier* := $H|\!)$ **using** *assms* **by** (*metis normal-restrict-supergroup*)
hence *grpHN*:*group* ($G(\!|$*carrier* := $H|\!)$ *Mod N*) **by** (*rule normal.factorgroup-is-group*)
have ($<\#>_{G(\!|carrier:=H|\!)}$) = ($\lambda U\ K.\ (\bigcup h \in U.\ \bigcup k \in K.\ \{h \otimes_{G(\!|carrier := H|\!)} k\})$) **using** *set-mult-def* **by** *metis*
moreover have ... = ($\lambda U\ K.\ (\bigcup h \in U.\ \bigcup k \in K.\ \{h \otimes_G k\})$) **by** *auto*
moreover have ($<\#>$) = ($\lambda U\ K.\ (\bigcup h \in U.\ \bigcup k \in K.\ \{h \otimes k\})$) **using** *set-mult-def*
by *metis*
ultimately have ($<\#>_{G(\!|carrier:=H|\!)}$) = ($<\#>_G$) **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 $NG{:}N \lhd G$ **and** $NH{:}N \subseteq H$ **and** $HG{:}H \lhd G$
shows $(rcosets_{G(\!|carrier := H|\!)}\ N) \lhd (G\ Mod\ N)$
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 $U <\#> V <\#> inv_{G\ Mod\ N}\ U = N\ \#>\ (g \otimes h \otimes inv\ g)$
proof −
from *g U* **have** $inv_{G\ Mod\ N}\ U = N\ \#>\ inv\ g$ **using** *NG normal.inv-FactGroup normal.rcos-inv* **by** *fastforce*
hence $U <\#> V <\#> inv_{G\ Mod\ N}\ U = (N\ \#>\ g) <\#> (N\ \#>\ h) <\#> (N\ \#>\ inv\ g)$ **using** *g h* **by** *simp*
also have $\ldots = N\ \#>\ (g \otimes h) <\#> (N\ \#>\ inv\ g)$ **using** *g hG NG normal.rcos-sum* **by** *force*
also have $\ldots = N\ \#>\ (g \otimes h \otimes inv\ g)$ **using** *g inv-closed ghG NG normal.rcos-sum* **by** *force*
finally show *?thesis* **.**
qed
ultimately show $U <\#> V <\#> inv_{G\ Mod\ N}\ U \in rcosets_{G(\!|carrier := H|\!)}\ N$ **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 Mod H*) = {$\mathbf{1}_{G\ Mod\ H}$}) = (*H* = *carrier G*)
proof
assume *carrier* (*G Mod H*) = {$\mathbf{1}_{G\ Mod\ H}$}
moreover with *assms lagrange* **have** *order* (*G Mod H*) ∗ *card H* = *order G* **unfolding** *FactGroup-def order-def* **using** *is-subgroup* **by** *force*
ultimately have *card H* = *order G* **unfolding** *order-def* **by** *auto*
thus *H* = *carrier G* **using** *subgroup.subset is-subgroup assms card-subset-eq* **unfolding** *order-def*
by *metis*
next
from *assms* **have** *ordergt0*:*order G* > *0* **unfolding** *order-def* **by** (*metis subgroup.finite-imp-card-positive subgroup-self*)
assume *H* = *carrier G*
hence *card H* = *order G* **unfolding** *order-def* **by** *simp*
with *assms is-subgroup lagrange* **have** *card* (*rcosets H*) ∗ *order G* = *order G* **by** *metis*
with *ordergt0* **have** *card* (*rcosets H*) = *1* **by** (*metis mult-eq-self-implies-10 mult.commute neq0-conv*)
hence *order* (*G Mod H*) = *1* **unfolding** *order-def FactGroup-def* **by** *auto*
thus *carrier* (*G Mod H*) = {$\mathbf{1}_{G\ Mod\ H}$} **using** *factorgroup-is-group* **by** (*metis group.order-one-triv-iff*)
qed

Finite groups have finite quotients.

lemma (**in** *normal*) *factgroup-finite*:
assumes *finite* (*carrier G*)
shows *finite* (*rcosets H*)
using *assms* **unfolding** *RCOSETS-def* **by** *auto*

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

lemma (**in** *normal*) *factgroup-subgroup-union-char*:
assumes *subgroup A* (*G Mod H*)
shows (⋃ *A*) = {*x* ∈ *carrier G*. *H* #> *x* ∈ *A*}
proof
show ⋃ *A* ⊆ {*x* ∈ *carrier G*. *H* #> *x* ∈ *A*}
proof
fix *x*
assume *x*:*x* ∈ ⋃ *A*
then obtain *a* **where** *a*:*a* ∈ *A x* ∈ *a* **by** *auto*
with *assms* **have** *xx*:*x* ∈ *carrier G* **using** *subgroup.subset* **unfolding** *FactGroup-def RCOSETS-def r-coset-def* **by** *force*
from *assms a* **obtain** *y* **where** *y*:*y* ∈ *carrier G a* = *H* #> *y* **using** *subgroup.subset* **unfolding** *FactGroup-def RCOSETS-def* **by** *force*
with *a* **have** *x* ∈ *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* ∈ *A* **by** *auto*

```
    with xx show x ∈ {x ∈ carrier G. H #> x ∈ A} by simp
  qed
next
  show {x ∈ carrier G. H #> x ∈ A} ⊆ ⋃A
  proof
    fix x
    assume x:x ∈ {x ∈ carrier G. H #> x ∈ A}
    hence xx:x ∈ carrier G H #> x ∈ A by auto
    moreover have x ∈ H #> x by (metis is-subgroup rcos-self xx(1))
    ultimately show x ∈ ⋃A by auto
  qed
qed

lemma (in normal) factgroup-subgroup-union-subgroup:
  assumes subgroup A (G Mod H)
  shows subgroup (⋃A) G
proof −
  have subgroup {x ∈ carrier G. H #> x ∈ A} G
  proof
    show {x ∈ carrier G. H #> x ∈ A} ⊆ carrier G by auto
  next
    fix x y
    assume x ∈ {x ∈ carrier G. H #> x ∈ A} and y ∈ {x ∈ carrier G. H #> x
∈ A}
    hence x:x ∈ carrier G H #> x ∈ A and y:y ∈ carrier G H #> y ∈ A by auto
    hence xyG:x ⊗ y ∈ carrier G by (metis m-closed)
    from assms x y have (H #> x) <#> (H #> y) ∈ A using subgroup.m-closed
unfolding FactGroup-def by fastforce
    hence H #> (x ⊗ y) ∈ A by (metis rcos-sum x(1) y(1))
    with xyG show x ⊗ y ∈ {x ∈ carrier G. H #> x ∈ A} by simp
  next
    have H #> 1 ∈ A using assms subgroup.one-closed unfolding FactGroup-def
by (metis coset-mult-one monoid.select-convs(2) subset)
    with assms one-closed show 1 ∈ {x ∈ carrier G. H #> x ∈ A} by simp
  next
    fix x
    assume x ∈ {x ∈ carrier G. H #> x ∈ A}
    hence x:x ∈ carrier G H #> x ∈ A by auto
    hence invx:inv x ∈ carrier G using inv-closed by simp
     from assms x have set-inv (H #> x) ∈ A using subgroup.m-inv-closed by
(metis inv-FactGroup subgroup.mem-carrier)
    hence H #> (inv x) ∈ A by (metis rcos-inv x(1))
    with invx show inv x ∈ {x ∈ carrier G. H #> x ∈ A} by simp
  qed
  with assms factgroup-subgroup-union-char show ?thesis by auto
qed

lemma (in normal) factgroup-subgroup-union-normal:
  assumes A ◁ (G Mod H)
```

shows $\bigcup A \lhd G$
proof −
have $\{x \in carrier\ G.\ H\ \#>\ x \in A\} \lhd G$
unfolding *normal-def normal-axioms-def*
proof *auto*
from *assms* **show** *subgroup* $\{x \in carrier\ G.\ H\ \#>\ x \in A\}\ G$
by (*metis* (*full-types*) *factgroup-subgroup-union-char factgroup-subgroup-union-subgroup normal-imp-subgroup*)
next
interpret *Anormal*: *normal A* (*G Mod H*) **using** *assms* **by** *simp*
fix *x y*
assume *x*:$x \in carrier\ G\ y \in \{x \in carrier\ G.\ H\ \#>\ x \in A\}\ \#>\ x$
then obtain x' **where** $x' \in \{x \in carrier\ G.\ H\ \#>\ x \in A\}\ y = x' \otimes x$ **unfolding** *r-coset-def* **by** *auto*
hence x':$x' \in carrier\ G\ H\ \#>\ x' \in A$ **by** *auto*
from *x*(*1*) **have** *Hx*:$H\ \#>\ x \in carrier\ (G\ Mod\ H)$ **unfolding** *FactGroup-def RCOSETS-def* **by** *force*
with x' **have** $(inv_{G\ Mod\ H}\ (H\ \#>\ x)) \otimes_{G\ Mod\ H} (H\ \#>\ x') \otimes_{G\ Mod\ H} (H\ \#>\ x) \in A$ **using** *Anormal.inv-op-closed1* **by** *auto*
hence $(set\text{-}inv\ (H\ \#>\ x))\ <\#>\ (H\ \#>\ x')\ <\#>\ (H\ \#>\ x) \in A$ **using** *inv-FactGroup Hx* **unfolding** *FactGroup-def* **by** *auto*
hence $(H\ \#>\ (inv\ x))\ <\#>\ (H\ \#>\ x')\ <\#>\ (H\ \#>\ x) \in A$ **using** *x*(*1*) **by** (*metis rcos-inv*)
hence $(H\ \#>\ (inv\ x \otimes x'))\ <\#>\ (H\ \#>\ x) \in A$ **by** (*metis inv-closed rcos-sum x'*(*1*) *x*(*1*))
hence $H\ \#>\ (inv\ x \otimes x' \otimes x) \in A$ **by** (*metis inv-closed m-closed rcos-sum x'*(*1*) *x*(*1*))
moreover have $inv\ x \otimes x' \otimes x \in carrier\ G$ **using** *x x'* **by** (*metis inv-closed m-closed*)
ultimately have $inv\ x \otimes x' \otimes x \in \{x \in carrier\ G.\ H\ \#>\ x \in A\}$ **by** *auto*
hence *xcoset*:$x \otimes (inv\ x \otimes x' \otimes x) \in x\ <\#\ \{x \in carrier\ G.\ H\ \#>\ x \in A\}$ **unfolding** *l-coset-def* **using** *x*(*1*) **by** *auto*
have $x \otimes (inv\ x \otimes x' \otimes x) = (x \otimes inv\ x) \otimes x' \otimes x$ **by** (*metis Units-eq Units-inv-Units m-assoc m-closed x'*(*1*) *x*(*1*))
also have $\ldots = x' \otimes x$ **by** (*metis l-one r-inv x'*(*1*) *x*(*1*))
also have $\ldots = y$ **by** (*metis* ‹$y = x' \otimes x$›)
finally have $x \otimes (inv\ x \otimes x' \otimes x) = y$**.**
with *xcoset* **show** $y \in x\ <\#\ \{x \in carrier\ G.\ H\ \#>\ x \in A\}$ **by** *auto*
next
interpret *Anormal*: *normal A* (*G Mod H*) **using** *assms* **by** *simp*
fix *x y*
assume *x*:$x \in carrier\ G\ y \in x\ <\#\ \{x \in carrier\ G.\ H\ \#>\ x \in A\}$
then obtain x' **where** $x' \in \{x \in carrier\ G.\ H\ \#>\ x \in A\}\ y = x \otimes x'$ **unfolding** *l-coset-def* **by** *auto*
hence x':$x' \in carrier\ G\ H\ \#>\ x' \in A$ **by** *auto*
from *x*(*1*) **have** *invx*:$inv\ x \in carrier\ G$ **by** (*rule inv-closed*)
hence *Hinvx*:$H\ \#>\ (inv\ x) \in carrier\ (G\ Mod\ H)$ **unfolding** *FactGroup-def RCOSETS-def* **by** *force*
with x' **have** $(inv_{G\ Mod\ H}\ (H\ \#>\ inv\ x)) \otimes_{G\ Mod\ H} (H\ \#>\ x') \otimes_{G\ Mod\ H}$

$(H \#> inv\ x) \in A$ **using** *invx Anormal.inv-op-closed1* **by** *auto*
hence $(set\text{-}inv\ (H \#> inv\ x)) <\#> (H \#> x') <\#> (H \#> inv\ x) \in A$ **using** *inv-FactGroup Hinvx* **unfolding** *FactGroup-def* **by** *auto*
hence $(H \#> inv\ (inv\ x)) <\#> (H \#> x') <\#> (H \#> inv\ x) \in A$ **using** *invx* **by** (*metis rcos-inv*)
hence $(H \#> x) <\#> (H \#> x') <\#> (H \#> inv\ x) \in A$ **by** (*metis inv-inv x(1)*)
hence $(H \#> (x \otimes x')) <\#> (H \#> inv\ x) \in A$ **by** (*metis rcos-sum x'(1) x(1)*)
hence $H \#> (x \otimes x' \otimes inv\ x) \in A$ **by** (*metis inv-closed m-closed rcos-sum x'(1) x(1)*)
moreover have $x \otimes x' \otimes inv\ x \in carrier\ G$ **using** $x\ x'$ **by** (*metis inv-closed m-closed*)
ultimately have $x \otimes x' \otimes inv\ x \in \{x \in carrier\ G.\ H \#> x \in A\}$ **by** *auto*
hence $xcoset{:}(x \otimes x' \otimes inv\ x) \otimes x \in \{x \in carrier\ G.\ H \#> x \in A\} \#> x$ **unfolding** *r-coset-def* **using** *invx* **by** *auto*
have $(x \otimes x' \otimes inv\ x) \otimes x = (x \otimes x') \otimes (inv\ x \otimes x)$ **by** (*metis Units-eq Units-inv-Units m-assoc m-closed x'(1) x(1)*)
also have $\ldots = x \otimes x'$ **using** $x(1)$ *l-inv* $x'(1)$ *m-closed r-one* **by** *auto*
also have $\ldots = y$ **by** (*metis* $\langle y = x \otimes x' \rangle$)
finally have $x \otimes x' \otimes inv\ x \otimes x = y$**.**
with *xcoset* **show** $y \in \{x \in carrier\ G.\ H \#> x \in A\} \#> x$ **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 $A = rcosets_{G(\!|carrier := \bigcup A|\!)}\ H$
proof −
have $A = rcosets_{G(\!|carrier := \{x \in carrier\ G.\ H \#> x \in A\}|\!)}\ H$
proof *auto*
fix *U*
assume $U{:}U \in A$
then obtain x' **where** $x'{:}x' \in carrier\ G\ U = H \#> x'$ **using** *assms subgroup.subset* **unfolding** *FactGroup-def RCOSETS-def* **by** *force*
with *U* **have** $H \#> x' \in A$ **by** *simp*
with x' **show** $U \in rcosets_{G(\!|carrier := \{x \in carrier\ G.\ H \#> x \in A\}|\!)}\ H$ **unfolding** *RCOSETS-def r-coset-def* **by** *auto*
next
fix *U*
assume $U{:}U \in rcosets_{G(\!|carrier := \{x \in carrier\ G.\ H \#> x \in A\}|\!)}\ H$
then obtain x' **where** $x'{:}x' \in \{x \in carrier\ G.\ H \#> x \in A\}\ U = H \#> x'$ **unfolding** *RCOSETS-def r-coset-def* **by** *auto*
hence $x' \in carrier\ G\ H \#> x' \in A$ **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* = (|*carrier*=(*rep* ' (*carrier G*)),
monoid.mult=(λ *x y*. *rep* ((*the-inv-into* (*carrier G*) *rep x*) ⊗$_G$ (*the-inv-into* (*carrier G*) *rep y*))),
one=*rep* **1**$_G$ |)

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

lemma *flatten-set-group*:
assumes *group*:*group G*
assumes *inj*:*inj-on rep* (*carrier G*)
shows *group* (*flatten G rep*)
proof (*rule groupI*)
fix *x y*
assume *x*:*x* ∈ *carrier* (*flatten G rep*) **and** *y*:*y* ∈ *carrier* (*flatten G rep*)
define *g h*
where *g* = *the-inv-into* (*carrier G*) *rep x*
and *h* = *the-inv-into* (*carrier G*) *rep y*
hence *x* ⊗$_{flatten\ G\ rep}$ *y* = *rep* (*g* ⊗$_G$ *h*) **unfolding** *flatten-def* **by** *auto*
moreover from *g-def h-def* **have** *g* ∈ *carrier G h* ∈ *carrier G*
using *inj x y the-inv-into-into* **unfolding** *flatten-def* **by** (*metis partial-object.select-convs*(*1*) *subset-refl*)+
hence *g* ⊗$_G$ *h* ∈ *carrier G* **by** (*metis group group.is-monoid monoid.m-closed*)

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

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

lemma (**in** *normal*) *flatten-set-group-mod-inj*:
shows *inj-on* (λ*U*. *SOME g*. *g* ∈ *U*) (*carrier* (*G Mod H*))
proof (*rule inj-onI*)
fix *U V*
assume *U*:*U* ∈ *carrier* (*G Mod H*) **and** *V*:*V* ∈ *carrier* (*G Mod H*)
then obtain *g h* **where** *g*:*U* = *H* #> *g g* ∈ *carrier G* **and** *h*:*V* = *H* #> *h h*
∈ *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* ∈ *U*) = (*SOME g*. *g* ∈ *V*)
with *notempty* **have** (*SOME g*. *g* ∈ *U*) ∈ *U* ∩ *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*) (λ*U*. *SOME g*. *g* ∈ *U*))
using *factorgroup-is-group flatten-set-group-mod-inj* **by** (*rule flatten-set-group*)

lemma (**in** *normal*) *flatten-set-group-mod-iso*:
shows (λ*U*. *SOME g*. *g* ∈ *U*) ∈ *iso* (*G Mod H*) (*flatten* (*G Mod H*) (λ*U*. *SOME*
g. *g* ∈ *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* ◁ *G* ⟹ (*H* = *carrier G* ∨ *H* = {**1**})

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

Simple groups are non-trivial.

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

Every group of prime order is simple

lemma (**in** *group*) *prime-order-simple*:
assumes *prime*:*prime* (*order G*)
shows *simple-group G*
proof
from *prime* **show** *1* < *order G* **unfolding** *prime-nat-iff* **by** *auto*
next
fix *H*
assume *H* ◁ *G*
hence *HG*:*subgroup H G* **unfolding** *normal-def* **by** *simp*
hence *card H dvd order G* **by** (*rule card-subgrp-dvd*)
with *prime* **have** *card H* = *1* ∨ *card H* = *order G* **unfolding** *prime-nat-iff* **by** *simp*
thus *H* = *carrier G* ∨ *H* = {**1**}
proof
assume *card H* = *1*
moreover from *HG* **have 1** ∈ *H* **by** (*metis subgroup.one-closed*)
ultimately show *?thesis* **by** (*auto simp*: *card-Suc-eq*)
next
assume *card H* = *order G*
moreover from *HG* **have** *H* ⊆ *carrier G* **unfolding** *subgroup-def* **by** *simp*
moreover from *prime* **have** *card* (*carrier G*) > *1* **unfolding** *order-def prime-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*:$\varphi \in$ *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) $\in$ *iso H G* **using** *iso*
by (*simp add*: *iso-set-sym*)
fix *N*
assume *NH*:*N* $\lhd$ *H* **and** *Nneq1*:*N* $\neq$ {$\mathbf{1}_H$}
then interpret *Nnormal*: *normal N H* **by** *simp*
define *M* **where** *M* = (*inv-into* (*carrier G*) φ) ‘ *N*
hence *MG*:*M* $\lhd$ *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* $\neq$ {**1**}
proof(*rule notI*)
assume *M* = {**1**}
hence φ ‘ *M* = {φ **1**} **by** (*metis* (*full-types*) *image-empty image-insert*)
hence φ ‘ *M* = {$\mathbf{1}_H$} **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*:$H \neq$ *carrier* G
assumes *max-normal*:$\bigwedge J.\ J \lhd G \Longrightarrow J \neq H \Longrightarrow J \neq$ *carrier* $G \Longrightarrow \neg\ (H \subseteq J)$

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 = *simple-group* (G *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 factorgroup-is-group allI impI disjCI*)
from *finite factgroup-finite factorgroup-is-group group.finite-pos-order* **have** *gt0*:$0 <$ *card* (*rcosets* H)
unfolding *FactGroup-def order-def* **by** *force*
from *maxH.proper finite* **have** *carrier* (G *Mod* H) $\neq \{\mathbf{1}_{G\ Mod\ H}\}$ **using** *fact-group-trivial-iff* **by** *auto*
hence *1* $\neq$ *order* (G *Mod* H) **using** *factorgroup-is-group group.order-one-triv-iff* **by** *metis*
with *gt0* **show** *1* $<$ *order* (G *Mod* H) **unfolding** *order-def FactGroup-def* **by** *auto*
next
fix A'
assume A'*normal*:$A' \lhd G$ *Mod* H **and** A'*nottriv*:$A' \neq \{\mathbf{1}_{G\ Mod\ H}\}$
define A **where** $A = \bigcup A'$
have *A2*:$A \lhd G$ **using** A'*normal* **unfolding** *A-def* **by** (*rule factgroup-subgroup-union-normal*)
have $H \in A'$ **using** A'*normal normal-imp-subgroup subgroup.one-closed* **unfolding** *FactGroup-def* **by** *force*
hence $H \subseteq A$ **unfolding** *A-def* **by** *auto*
hence *A1*:$H \lhd (G(\!|carrier := A|\!))$ **using** *A2 is-normal* **by** (*metis is-subgroup maxH.max-normal normal-restrict-supergroup subgroup-self*)
have *A3*:$A' =$ *rcosets*$_{G(\!|carrier\ :=\ A|\!)}$ H

unfolding *A-def* **using** *factgroup-subgroup-union-factor A'normal normal-imp-subgroup* **by** *auto*
from *A1* **interpret** *normalHA*: *normal H* (*G*(|*carrier* := *A*|)) **by** *metis*
have $H \subseteq A$ **using** *normalHA.is-subgroup subgroup.subset* **by** *force*
with *A2* **have** $A = H \vee A =$ *carrier G* **using** *maxH.max-normal* **by** *auto*
thus $A' =$ *carrier* (*G Mod H*)
proof
assume $A = H$
hence *carrier* (*G*(|*carrier* := *A*|) *Mod H*) = $\{\mathbf{1}_{(G(|carrier := A|)\ Mod\ H)}\}$ **by** (*metis finite is-group normalHA.fact-group-trivial-iff normalHA.subgroup-self normalHA.subset subgroup-finite subgroup-of-restricted-group subgroup-of-subgroup subset-antisym*)
also have ... = $\{\mathbf{1}_{G\ Mod\ H}\}$ **unfolding** *FactGroup-def* **by** *auto*
finally have $A' = \{\mathbf{1}_{G\ Mod\ H}\}$ **using** *A3* **unfolding** *FactGroup-def* **by** *simp*
with *A'nottriv* **show** *?thesis*..
next
assume $A =$ *carrier G*
hence (*G*(|*carrier* := *A*|) *Mod H*) = *G Mod H* **by** *auto*
thus $A' =$ *carrier* (*G Mod H*) **using** *A3* **unfolding** *FactGroup-def* **by** *simp*
qed
qed
next
assume *simple*:*simple-group* (*G Mod H*)
show *max-normal-subgroup H G*
proof
from *simple* **have** *carrier* (*G Mod H*) $\neq \{\mathbf{1}_{G\ Mod\ H}\}$ **unfolding** *simple-group-def simple-group-axioms-def order-def* **by** *auto*
with *finite fact-group-trivial-iff* **show** $H \neq$ *carrier G* **by** *auto*
next
fix *A*
assume *A*:$A \lhd G$ $A \neq H$ $A \neq$ *carrier G*
show $\neg\ H \subseteq A$
proof
assume *HA*:$H \subseteq A$
hence $H \lhd$ (*G*(|*carrier* := *A*|)) **by** (*metis A(1) inv-op-closed2 is-subgroup normal-inv-iff normal-restrict-supergroup*)
then interpret *normalHA*: *normal H* (*G*(|*carrier* := *A*|)) **by** *simp*
from *finite* **have** *finiteA*:*finite A* **using** *A(1) normal-imp-subgroup* **by** (*metis subgroup-finite*)
have $rcosets_{(G(|carrier := A|))}$ $H \lhd G\ Mod\ H$ **using** *normality-factorization is-normal HA A(1)* **by** *auto*
with *simple* **have** $rcosets_{(G(|carrier := A|))}$ $H = \{\mathbf{1}_{G\ Mod\ H}\} \vee rcosets_{(G(|carrier := A|))}$ $H =$ *carrier* (*G Mod H*)
unfolding *simple-group-def simple-group-axioms-def* **by** *auto*
thus *False*
proof
assume $rcosets_{G(|carrier := A|)}$ $H = \{\mathbf{1}_{G\ Mod\ H}\}$
hence $rcosets_{G(|carrier := A|)}$ $H = \{\mathbf{1}_{(G(|carrier := A|))\ Mod\ H}\}$ **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*:$rcosets_{G(\!|carrier := A|\!)}\ H = carrier\ (G\ Mod\ H)$
have $A = carrier\ G$ **unfolding** *FactGroup-def RCOSETS-def*
proof
show $A \subseteq carrier\ G$ **using** *A(1) normal-imp-subgroup subgroup.subset* **by** *metis*
next
show $carrier\ G \subseteq A$
proof
fix x
assume x:$x \in carrier\ G$
hence $H\ \#>\ x \in rcosets\ H$ **unfolding** *RCOSETS-def* **by** *auto*
with *AHGH* **have** $H\ \#>\ x \in rcosets_{G(\!|carrier := A|\!)}\ H$ **unfolding** *FactGroup-def* **by** *simp*
then obtain x' **where** x':$x' \in A\ H\ \#>x = H\ \#>_{G(\!|carrier := A|\!)}\ x'$ **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 ∃!*Q*. *Q* ∈ *subgroups-of-size q*
shows (*THE Q*. *Q* ∈ *subgroups-of-size q*) ◁ *G*
proof −
from *assms* **obtain** *Q* **where** *Q* ∈ *subgroups-of-size q* **by** *auto*
define *Q* **where** *Q* = (*THE Q*. *Q* ∈ *subgroups-of-size q*)
with *assms* **have** *Qsize*:*Q* ∈ *subgroups-of-size q* **using** *theI* **by** *metis*
hence *QG*:*subgroup Q G* **and** *cardQ*:*card Q* = *q* **unfolding** *subgroups-of-size-def*
by *auto*
from *QG* **have** *Q* ◁ *G* **apply**(*rule normalI*)
proof
fix *g*
assume *g*:*g* ∈ *carrier G*
hence *invg*:*inv g* ∈ *carrier G* **by** (*metis inv-closed*)
with *fin Qsize* **have** *conjugation-action q* (*inv g*) *Q* ∈ *subgroups-of-size q* **by**
(*metis conjugation-is-size-invariant*)
with *g Qsize* **have** (*inv g*) <# (*Q* #> *inv* (*inv g*)) ∈ *subgroups-of-size q*
unfolding *conjugation-action-def* **by** *auto*
with *invg g* **have** *inv g* <# (*Q* #> *g*) = *Q* **by** (*metis Qsize assms*(*2*) *inv-inv*)
with *QG QG g* **show** *Q* #> *g* = *g* <# *Q* **by** (*rule conj-wo-inv*)
qed
with *Q-def* **show** *?thesis* **by** *simp*
qed

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

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

moreover have *card* (*subgroups-of-size q*) *dvd p* **by** (*metis power-one-right*
syl.num-sylow-dvd-remainder)

then have *card* (*subgroups-of-size q*) = *p* ∨ *card* (*subgroups-of-size q*) = *1*
using *primep* **by** (*auto simp add*: *prime-nat-iff*)
ultimately have *card* (*subgroups-of-size q*) = *1* **using** *pq*
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*) ◁ *G*
using *assms* **by** (*metis pq-order-unique-subgrp unique-sizes-subgrp-normal*)

The trivial subgroup is normal in every group.

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

8.2 Normal Series

We define a normal series as a locale which fixes one group G and a list $\mathfrak{G}$ 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* $\mathfrak{G}$ = *carrier G*
assumes *normal*:$\bigwedge i$. *i* + *1* < *length* $\mathfrak{G} \Longrightarrow$ ($\mathfrak{G}$! *i*) ◁ *G*(|*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**}, *carrier G*]
unfolding *normal-series-def normal-series-axioms-def*
using *is-group trivial-subgroup-is-normal* **by** *auto*

We can also show that the normal series presented above is the only such with a length of two:

```
lemma (in normal-series) length-two-unique:
  assumes length 𝔊 = 2
  shows 𝔊 = [{1}, carrier G]
proof(rule nth-equalityI)
  from assms show length 𝔊 = length [{1}, carrier G] by auto
next
  show 𝔊 ! i = [{1}, carrier G] ! i if i:i < length 𝔊 for i
  proof −
    have i = 0 ∨ i = 1 using that assms by auto
    thus 𝔊 ! i = [{1}, carrier G] ! i
    proof(rule disjE)
      assume i:i = 0
      hence 𝔊 ! i = hd 𝔊 by (metis hd-conv-nth notempty)
      thus 𝔊 ! i = [{1}, carrier G] ! i using hd i by simp
    next
      assume i:i = 1
       with assms have 𝔊 ! i = last 𝔊 by (metis diff-add-inverse last-conv-nth
nat-1-add-1 notempty)
      thus 𝔊 ! i = [{1}, 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 $H \lhd G$ we receive a normal series for G.

```
lemma (in group) normal-series-extend:
  assumes normal:normal-series (G(|carrier := H|)) 𝔎
  assumes HG:H ◁ G
  shows normal-series G (𝔎 @ [carrier G])
proof −
  from normal interpret normalH: normal-series (G(|carrier := H|)) 𝔎.
  from normalH.hd have hd 𝔎 = {1} by simp
   with normalH.notempty have hdTriv:hd (𝔎 @ [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 ∈ hd (𝔎 @ [carrier G])
    with hdTriv show x = 1 by simp
  next
    from hdTriv show  1 ∈ hd (𝔎 @ [carrier G]) by simp
  next
    fix i
    assume i:i < length 𝔎
    show (𝔎 @ [carrier G]) ! i ◁ G(|carrier := (𝔎 @ [carrier G]) ! Suc i|)
    proof (cases i + 1 < length 𝔎)
      case True
      with normalH.normal have 𝔎 ! i ◁ G(|carrier := 𝔎 ! (i + 1)|) by auto
```

```
      with i have (𝔎 @ [carrier G]) ! i ◁ G(|carrier := 𝔎 ! (i + 1)|) using
nth-append by metis
      with True show (𝔎 @ [carrier G]) ! i ◁ G(|carrier := (𝔎 @ [carrier G]) !
(Suc i)|) using nth-append Suc-eq-plus1 by metis
    next
      case False
      with i have i2:i + 1 = length 𝔎 by simp
      from i have (𝔎 @ [carrier G]) ! i = 𝔎 ! i by (metis nth-append)
    also from i2 normalH.notempty have ... = last 𝔎 by (metis add-diff-cancel-right'
last-conv-nth)
      also from normalH.last have ... = H by simp
      finally have (𝔎 @ [carrier G]) ! i = H.
      moreover from i2 have (𝔎 @ [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 𝔊 ⟹ subgroup (𝔊 ! i) G
proof −
  have i + 1 < length 𝔊 ⟹ subgroup (𝔊 ! i) G
  proof (induction length 𝔊 − (i + 2) arbitrary: i)
    case 0
    hence i:i + 2 = length 𝔊 by simp
    hence ii:i + 1 = length 𝔊 − 1 by force
    from i normal have 𝔊 ! i ◁ G(|carrier := 𝔊 ! (i + 1)|) by auto
      with ii last notempty show subgroup (𝔊 ! i) G using last-conv-nth nor-
mal-imp-subgroup by fastforce
  next
    case (Suc k)
     from Suc(3) normal have i:subgroup (𝔊 ! i) (G(|carrier := 𝔊 ! (i + 1)|))
using normal-imp-subgroup by auto
    from Suc(2) have k:k = length 𝔊 − ((i + 1) + 2) by arith
    with Suc have subgroup (𝔊 ! (i + 1)) G by simp
   with i show subgroup (𝔊 ! i) G by (metis is-group subgroup.subgroup-of-subgroup)
  qed
  moreover have i + 1 = length 𝔊 ⟹ subgroup (𝔊 ! i) G
   using last notempty last-conv-nth by (metis add-diff-cancel-right' subgroup-self)
  ultimately show i < length 𝔊 ⟹ subgroup (𝔊 ! i) G by force
qed
```

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

```
lemma (in normal-series) normal-series-snd-to-last:
  shows 𝔊 ! (length 𝔊 − 2) ◁ G
proof (cases 2 ≤ length 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** $\mathfrak{G}$! (*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 (*length* $\mathfrak{G} - 2$) + 1 < *length* $\mathfrak{G}$ **by** *arith*
with *normal last* **have** $\mathfrak{G}$! (*length* $\mathfrak{G} - 2$) ◁ $G(\!|carrier := \mathfrak{G}$! ((*length* $\mathfrak{G} - 2$) + 1)$|\!)$ **by** *auto*
have 1 + (1 + (*length* $\mathfrak{G} - (1 + 1)$)) = *length* $\mathfrak{G}$
using *True le-add-diff-inverse* **by** *presburger*
then have $\mathfrak{G}$! (*length* $\mathfrak{G} - 2$) ◁ $G(\!|carrier :=$ $\mathfrak{G}$! (*length* $\mathfrak{G} - 1$)$|\!)$
by (*metis* ‹$\mathfrak{G}$! (*length* $\mathfrak{G} - 2$) ◁ G ($\!|carrier := \mathfrak{G}$! (*length* $\mathfrak{G} - 2 + 1$)$|\!$)› *add.commute add-diff-cancel-left' one-add-one*)
with *notempty last* **show** *?thesis* **using** *last-conv-nth* **by** *force*
qed

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

lemma (**in** *normal-series*) *normal-series-prefix-closed*:
assumes $i \leq$ *length* $\mathfrak{G}$ **and** $0 < i$
shows *normal-series* ($G(\!|carrier := \mathfrak{G}$! ($i - 1$)$|\!)$) (*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**}, (*THE H*. $H \in$ *subgroups-of-size q*), *carrier G*]
proof −
define *H* **where** *H* = (*THE H*. $H \in$ *subgroups-of-size q*)
with *assms* **have** *HG*:$H \lhd G$ **by** (*metis pq-order-subgrp-normal*)
then interpret *groupH*: *group* $G(\!|carrier := H|\!)$ **unfolding** *normal-def* **by** (*metis subgroup-imp-group*)
have *normal-series* ($G(\!|carrier := H|\!)$) [{**1**}, *H*] **using** *groupH.trivial-normal-series* **by** *auto*
with *HG* **show** *?thesis* **unfolding** *H-def* **by** (*metis append-Cons append-Nil normal-series-extend*)
qed

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

definition (**in** *normal-series*) *quotients*
  where *quotients* = *map* (λ*i*. *G*(|*carrier* := 𝔊 ! (*i* + *1*)|) *Mod* 𝔊 ! *i*) [*0*..<((*length* 𝔊) − *1*)]

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

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

lemma (**in** *normal-series*) *last-quotient*:
  assumes *length* 𝔊 > *1*
  shows *last quotients* = *G Mod* 𝔊 ! (*length* 𝔊 − *1* − *1*)
proof −
  from *assms* **have** *lsimp*:*length* 𝔊 − *1* − *1* + *1* = *length* 𝔊 − *1* **by** *auto*
  from *assms* **have** *quotients* ≠ [] **unfolding** *quotients-def* **by** *auto*
  hence *last quotients* = *quotients* ! (*length quotients* − *1*) **by** (*metis last-conv-nth*)
  also have ... = *quotients* ! (*length* 𝔊 − *1* − *1*) **by** (*metis add-diff-cancel-left' quotients-length add.commute*)
  also have ... = *G*(|*carrier* := 𝔊 ! ((*length* 𝔊 − *1* − *1*) + *1*)|) *Mod* 𝔊 ! (*length* 𝔊 − *1* − *1*)
    unfolding *quotients-def* **using** *assms* **by** *auto*
  also have ... = *G*(|*carrier* := 𝔊 ! (*length* 𝔊 − *1*)|) *Mod* 𝔊 ! (*length* 𝔊 − *1* − *1*) **using** *lsimp* **by** *simp*
  also have ... = *G Mod* 𝔊 ! (*length* 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*:*group H*
  assumes *iso*:Ψ ∈ *iso G H*
  shows *normal-series H* (*map* (*image* Ψ) 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* Ψ **unfolding** *group-hom-def group-hom-axioms-def iso-def* **by** *auto*
  have *hd* (*map* (*image* Ψ) 𝔊) = Ψ ‘ {**1**} **by** (*metis hd-map hd notempty*)
  also have ... = {Ψ **1**} **by** (*metis image-empty image-insert*)
  also have ... = {**1**$_H$} **using** *group-hom group-hom.hom-one* **by** *auto*

finally show *hd* (*map* ((`) Ψ) 𝔊) = {**1**$_H$}.
next
show *last* (*map* ((`) Ψ) 𝔊) = *carrier H* ∧ (∀ *i*. *Suc i* < *length* 𝔊 ⟶ Ψ ` 𝔊 ! *i* ◁ *H*⦇*carrier* := Ψ ` 𝔊 ! *Suc i*⦈)
proof (*auto del*: *equalityI*)
have *last* (*map* ((`) Ψ) 𝔊) = Ψ ` (*carrier G*) **using** *last last-map notempty* **by** *metis*
also have ... = *carrier H* **using** *iso* **unfolding** *iso-def bij-betw-def* **by** *simp*
finally show *last* (*map* ((`) Ψ) 𝔊) = *carrier H*.
next
fix *i*
assume *i*:*Suc i* < *length* 𝔊
hence *norm*:𝔊 ! *i* ◁ *G*⦇*carrier* := 𝔊 ! *Suc i*⦈ **using** *normal* **by** *simp*
moreover have *restrict* Ψ (𝔊 ! *Suc i*) ∈ *iso* (*G*⦇*carrier* := 𝔊 ! *Suc i*⦈) (*H*⦇*carrier* := Ψ ` 𝔊 ! *Suc i*⦈)
by (*metis H i is-group iso iso-restrict normal-series-subgroups*)
moreover have *group* (*G*⦇*carrier* := 𝔊 ! *Suc i*⦈) **by** (*metis i normal-series-subgroups subgroup-imp-group*)
moreover hence *subgroup* (𝔊 ! *Suc i*) *G* **by** (*metis i normal-series-subgroups*)
hence *subgroup* (Ψ ` 𝔊 ! *Suc i*) *H* **by** (*metis H is-group iso iso-subgroup*)
hence *group* (*H*⦇*carrier* := Ψ ` 𝔊 ! *Suc i*⦈) **by** (*metis H subgroup.subgroup-is-group*)
ultimately have *restrict* Ψ (𝔊 ! *Suc i*) ` 𝔊 ! *i* ◁ *H*⦇*carrier* := Ψ ` 𝔊 ! *Suc i*⦈
using *is-group H iso-normal-subgroup* **by** (*auto cong del*: *image-cong-simp*)
moreover from *norm* **have** 𝔊 ! *i* ⊆ 𝔊 ! *Suc i* **unfolding** *normal-def subgroup-def* **by** *auto*
hence {*y*. ∃ *x*∈𝔊 ! *i*. *y* = (*if x* ∈ 𝔊 ! *Suc i then* Ψ *x else undefined*)} = {*y*. ∃ *x*∈𝔊 ! *i*. *y* = Ψ *x*} **by** *auto*
ultimately show Ψ ` 𝔊 ! *i* ◁ *H*⦇*carrier* := Ψ ` 𝔊 ! *Suc i*⦈ **unfolding** *restrict-def image-def* **by** *auto*
qed
qed

8.3 Composition Series

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

locale *composition-series* = *normal-series* +
assumes *simplefact*:⋀*i*. *i* + *1* < *length* 𝔊 ⟹ *simple-group* (*G*⦇*carrier* := 𝔊 ! (*i* + *1*)⦈ *Mod* 𝔊 ! *i*)

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

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

lemma (**in** *composition-series*) *composition-series-length-one*:
shows (*length* 𝔊 = *1*) = (𝔊 = [{**1**}])

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

lemma (**in** *composition-series*) *composition-series-triv-group*:
shows (*carrier* $G = \{\mathbf{1}\}$) = ($\mathfrak{G} = [\{\mathbf{1}\}]$)
proof
assume *G*:*carrier* $G = \{\mathbf{1}\}$
have *length* $\mathfrak{G} = 1$
proof (*rule ccontr*)
assume *length* $\mathfrak{G} \neq 1$
with *notempty* **have** *length*:*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-conv-nth notempty* **have** *simple-group* ($G(\!|$*carrier* := $\mathfrak{G}$! *1*$|\!)$ *Mod* $\{\mathbf{1}\}$) **by** *force*
moreover have *SG*:*subgroup* ($\mathfrak{G}$! *1*) *G* **using** *length normal-series-subgroups* **by** *auto*
hence *group* ($G(\!|$*carrier* := $\mathfrak{G}$! *1*$|\!)$) **by** (*metis subgroup-imp-group*)
ultimately have *simple-group* ($G(\!|$*carrier* := $\mathfrak{G}$! *1*$|\!)$) **using** *group.trivial-factor-iso simple-group.iso-simple* **by** *fastforce*
moreover from *SG G* **have** *carrier* ($G(\!|$*carrier* := $\mathfrak{G}$! *1*$|\!)$) = $\{\mathbf{1}\}$ **unfolding** *subgroup-def* **by** *auto*
ultimately show *False* **using** *simple-group.simple-not-triv* **by** *force*
qed
thus $\mathfrak{G} = [\{\mathbf{1}\}]$ **by** (*metis composition-series-length-one*)
next
assume $\mathfrak{G} = [\{\mathbf{1}\}]$
with *last* **show** *carrier* $G = \{\mathbf{1}\}$ **by** *auto*
qed

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

lemma (**in** *composition-series*) *inner-elements-not-triv*:
assumes $i + 1 <$ *length* $\mathfrak{G}$
assumes $i > 0$
shows $\mathfrak{G}$! $i \neq \{\mathbf{1}\}$
proof
from *assms* **have** $(i - 1) + 1 <$ *length* $\mathfrak{G}$ **by** *simp*
hence *simple*:*simple-group* ($G(\!|$*carrier* := $\mathfrak{G}$! $((i - 1) + 1)|\!)$ *Mod* $\mathfrak{G}$! $(i - 1)$) **using** *simplefact* **by** *auto*
assume *i*:$\mathfrak{G}$! $i = \{\mathbf{1}\}$
moreover from *assms* **have** $(i - 1) + 1 = i$ **by** *auto*
ultimately have $G(\!|$*carrier* := $\mathfrak{G}$! $((i - 1) + 1)|\!)$ *Mod* $\mathfrak{G}$! $(i - 1) = G(\!|$*carrier*

:= {**1**}⦈ *Mod* 𝔊 ! (*i* − *1*) **using** *i* **by** *auto*
hence *order* (*G*⦇*carrier* := 𝔊 ! ((*i* − *1*) + *1*)⦈ *Mod* 𝔊 ! (*i* − *1*)) = *1* **unfolding** *FactGroup-def order-def RCOSETS-def* **by** *force*
thus *False* **using** *i simple* **unfolding** *simple-group-def simple-group-axioms-def* **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*) = (𝔊 = [{**1**}, *carrier G*])
proof
assume 𝔊 = [{**1**}, *carrier G*]
with *simplefact* **have** *simple-group* (*G Mod* {**1**}) **by** *auto*
moreover have *the-elem* ∈ *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* 𝔊 > *1*
proof (*rule ccontr*)
assume ¬ *1* < *length* 𝔊
hence *length* 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* 𝔊 ≤ *2*
proof (*rule ccontr*)
define *k* **where** *k* = *length* 𝔊 − *2*
assume ¬ (*length* 𝔊 ≤ *2*)
hence *gt2*:*length* 𝔊 > *2* **by** *simp*
hence *ksmall*:*k* + *1* < *length* 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* ‹¬ *length* 𝔊 ≤ *2*› *add.commute nat-le-linear one-add-one*)
from *normal ksmall* **have** 𝔊 ! *k* ◁ *G*⦇ *carrier* := 𝔊 ! (*k* + *1*)⦈ **by** *simp*
from *simplefact ksmall* **have** *simplek*:*simple-group* (*G*⦇*carrier* := 𝔊 ! (*k* + *1*)⦈ *Mod* 𝔊 ! *k*) **by** *simp*
from *simplefact ksmall* **have** *simplek'*:*simple-group* (*G*⦇*carrier* := 𝔊 ! ((*k* − *1*) + *1*)⦈ *Mod* 𝔊 ! (*k* − *1*)) **by** *auto*
have 𝔊 ! *k* ◁ *G* **using** *carrier k-def gt2 normal ksmall* **by** *force*
with *simple* **have** (𝔊 ! *k*) = *carrier G* ∨ (𝔊 ! *k*) = {**1**} **unfolding** *simple-group-def simple-group-axioms-def* **by** *simp*
thus *False*
proof (*rule disjE*)
assume 𝔊 ! *k* = *carrier G*
hence *G*⦇*carrier* := 𝔊 ! (*k* + *1*)⦈ *Mod* 𝔊 ! *k* = *G Mod* (*carrier G*) **using**

carrier **by** *auto*
with *simplek self-factor-not-simple* **show** *False* **by** *auto*
next
assume $\mathfrak{G}$! $k = \{\mathbf{1}\}$
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 $\mathfrak{G} = [\{\mathbf{1}\}, carrier\ G]$ **by** (*rule length-two-unique*)
qed

Two consecutive elements in a composition series are distinct.

lemma (**in** *composition-series*) *entries-distinct*:
assumes *finite*:*finite* (*carrier G*)
assumes *i*:$i + 1 < length\ \mathfrak{G}$
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* $(carrier\ (G(\!|carrier := \mathfrak{G}$! $(i + 1)|\!|)))$ **by** *auto*
from *i* **have** *norm*:$\mathfrak{G}$! $i \lhd (G(\!|carrier := \mathfrak{G}$! $(i + 1)|\!|)$ **by** (*rule normal*)
assume $\mathfrak{G}$! $i = \mathfrak{G}$! $(i + 1)$
hence $\mathfrak{G}$! $i = carrier\ (G(\!|carrier := \mathfrak{G}$! $(i + 1)|\!|)$ **by** *auto*
hence $carrier\ ((G(\!|carrier := (\mathfrak{G}$! $(i + 1))|\!|)\ Mod\ (\mathfrak{G}$! $i)) = \{\mathbf{1}_{(G(\!|carrier := \mathfrak{G}\ !\ (i + 1)|\!|)\ Mod\ \mathfrak{G}\ !\ i}\}$
using *norm fin normal.fact-group-trivial-iff* **by** *metis*
hence ¬ *simple-group* $((G(\!|carrier := (\mathfrak{G}$! $(i + 1))|\!|)\ Mod\ (\mathfrak{G}$! $i))$ **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* $[\{\mathbf{1}\}, (THE\ H.\ H \in subgroups\text{-}of\text{-}size\ q), carrier\ G]$
unfolding *composition-series-def composition-series-axioms-def*
apply(*auto*)
using *assms* **apply**(*rule pq-order-normal-series*)
proof −
define *H* **where** $H = (THE\ H.\ H \in subgroups\text{-}of\text{-}size\ q)$
from *assms* **have** *exi*:$\exists!Q.\ Q \in (subgroups\text{-}of\text{-}size\ q)$ **by** (*auto simp*: *pq-order-unique-subgrp*)
hence *Hsize*:$H \in subgroups\text{-}of\text{-}size\ q$ **unfolding** *H-def* **using** *theI'* **by** *metis*
hence *HsubG*:*subgroup H G* **unfolding** *subgroups-of-size-def* **by** *auto*
then interpret *Hgroup*: *group* $G(\!|carrier := H|\!|)$ **by** (*metis subgroup-imp-group*)
fix *i*
assume $i < Suc\ (Suc\ 0)$
hence $i = 0 \vee i = 1$ **by** *auto*
thus *simple-group* $(G(\!|carrier := [H, carrier\ G]\ !\ i|\!|)\ Mod\ [\{\mathbf{1}\}, H, carrier\ G]\ !\ i)$

proof
assume *i*:*i* = *0*
from *Hsize* **have** *orderH*:*order* (*G*(|*carrier* := *H*|)) = *q* **unfolding** *subgroups-of-size-def order-def* **by** *simp*
hence *order* (*G*(|*carrier* := *H*|) *Mod* {**1**}) = *q* **unfolding** *FactGroup-def* **using** *card-rcosets-triv order-def*
by (*metis Hgroup.card-rcosets-triv HsubG finite monoid.cases-scheme monoid.select-convs*(*2*) *partial-object.select-convs*(*1*) *partial-object.update-convs*(*1*) *subgroup-finite*)
have *normal* {**1**} (*G*(|*carrier* := *H*|)) **by** (*metis Hgroup.is-group Hgroup.normal-inv-iff HsubG group.trivial-subgroup-is-normal is-group singleton-iff subgroup.one-closed subgroup.subgroup-of-subgroup*)
hence *group* (*G*(|*carrier* := *H*|) *Mod* {**1**}) **by** (*metis normal.factorgroup-is-group*)
with *orderH primeq* **have** *simple-group* (*G*(|*carrier* := *H*|) *Mod* {**1**}) **by** (*metis* ‹*order* (*G*(|*carrier* := *H*|) *Mod* {**1**}) = *q*› *group.prime-order-simple*)
with *i* **show** *?thesis* **by** *simp*
next
assume *i*:*i* = *1*
from *assms exi* **have** *H* ◁ *G* **unfolding** *H-def* **by** (*metis pq-order-subgrp-normal*)
hence *groupGH*:*group* (*G Mod H*) **by** (*metis normal.factorgroup-is-group*)
from *primeq* **have** *q* ≠ *0* **by** (*metis not-prime-0*)
from *HsubG finite orderG* **have** *card* (*rcosets H*) ∗ *card H* = *q* ∗ *p* **unfolding** *subgroups-of-size-def* **using** *lagrange* **by** *simp*
with *Hsize* **have** *card* (*rcosets H*) ∗ *q* = *q* ∗ *p* **unfolding** *subgroups-of-size-def* **by** *simp*
with ‹*q* ≠ *0*› **have** *card* (*rcosets H*) = *p* **by** *auto*
hence *order* (*G Mod H*) = *p* **unfolding** *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* ≤ *length* 𝔊 **and** *0* < *i*
shows *composition-series* (*G*(|*carrier* := 𝔊 ! (*i* − *1*)|)) (*take i* 𝔊)
unfolding *composition-series-def composition-series-axioms-def*
proof *auto*
from *assms* **show** *normal-series* (*G*(|*carrier* := 𝔊 ! (*i* − *Suc 0*)|)) (*take i* 𝔊) **by** (*metis One-nat-def normal-series-prefix-closed*)
next
fix *j*
assume *j*:*Suc j* < *length* 𝔊 *Suc j* < *i*
with *simplefact* **show** *simple-group* (*G*(|*carrier* := 𝔊 ! *Suc j*|) *Mod* 𝔊 ! *j*) **by** (*metis Suc-eq-plus1*)
qed

The second element in a composition series is simple group.

lemma (**in** *composition-series*) *composition-series-snd-simple*:
assumes *2* ≤ *length* 𝔊

shows *simple-group* (*G*(|*carrier* := 𝔊 ! *1*|))
proof −
from *assms* **interpret** *compTake*: *composition-series G*(|*carrier* := 𝔊 ! *1*|) *take 2* 𝔊 **by** (*metis add-diff-cancel-right′ composition-series-prefix-closed one-add-one zero-less-numeral*)
from *assms* **have** *length* (*take 2* 𝔊) = *2* **by** (*metis add-diff-cancel-right′ append-take-drop-id diff-diff-cancel length-append length-drop*)
hence (*take 2* 𝔊) = [{**1**$_{(G(|carrier := 𝔊 ! 1|))}$}, *carrier* (*G*(|*carrier* := 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* < *length* 𝔊
shows (*simple-group* (*G*(|*carrier* := 𝔊 ! *i*|))) = (*i* = *1*)
proof
assume *simpi*:*simple-group* (*G*(|*carrier* := 𝔊 ! *i*|))
hence 𝔊 ! *i* ≠ {**1**} **using** *simple-group.simple-not-triv* **by** *force*
hence *i* ≠ *0* **using** *hd hd-conv-nth notempty* **by** *auto*
then interpret *compTake*: *composition-series G*(|*carrier* := 𝔊 ! *i*|) *take* (*Suc i*) 𝔊
using *assms composition-series-prefix-closed* **by** (*metis diff-Suc-1 less-eq-Suc-le zero-less-Suc*)
from *simpi* **have** (*take* (*Suc i*) 𝔊) = [{**1**$_{G(|carrier := 𝔊 ! i|)}$}, *carrier* (*G*(|*carrier* := 𝔊 ! *i*|))]
by (*metis compTake.composition-series-simple-group*)
hence *length* (*take* (*Suc i*) 𝔊) = *2* **by** *auto*
hence *min* (*length* 𝔊) (*Suc i*) = *2* **by** (*metis length-take*)
with *assms* **have** *Suc i* = *2* **by** *force*
thus *i* = *1* **by** *simp*
next
assume *i*:*i* = *1*
with *assms* **have** *2* ≤ *length* 𝔊 **by** *simp*
with *i* **show** *simple-group* (*G*(|*carrier* := 𝔊 ! *i*|)) **by** (*metis composition-series-snd-simple*)
qed

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

lemma (**in** *composition-series*) *snd-to-last-max-normal*:
assumes *finite*:*finite* (*carrier G*)
assumes *length*:*length* 𝔊 > *1*
shows *max-normal-subgroup* (𝔊 ! (*length* 𝔊 − *2*)) *G*
unfolding *max-normal-subgroup-def max-normal-subgroup-axioms-def*
proof (*auto del*: *equalityI*)
show 𝔊 ! (*length* 𝔊 − *2*) ◁ *G* **by** (*rule normal-series-snd-to-last*)
next

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

```
RCOSETS-def by auto
        hence G' #> x ∈ rcosets_{G(|carrier := J|)} G' using facts-eq by auto
       then obtain j where j:j ∈ J G' #> x = G' #> j unfolding RCOSETS-def
r-coset-def by force
        hence x ∈ G' #> j using G'G normal-imp-subgroup x repr-independenceD
by fastforce
         then obtain g' where g':g' ∈ G' x = g' ⊗ j unfolding r-coset-def by
auto
        hence g' ∈ J using G'J normal-imp-subgroup subgroup.subset by force
           with g'(2) j(1) show  x ∈ 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 < length (remdups-adj xs) length xs > 0
  obtains j where j + 1 < length xs
    (remdups-adj xs) ! i = xs ! j (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 ≠ []
    by auto
  then obtain y xs' where xs: xs = y # xs'
    by (cases xs) blast
  from ‹xs ≠ []› have lenxs:length xs > 0 by simp
  from xs have rem:remdups-adj (x # xs) = (if x = y then remdups-adj (y # xs')
else x # remdups-adj (y # xs')) using remdups-adj.simps(3) by auto
  show thesis
  proof (cases x = y)
    case True
    with rem xs have rem2:remdups-adj (x # xs) = remdups-adj xs by auto
    with Cons(3) have i + 1 < length (remdups-adj xs) by simp
    with Cons.IH lenxs obtain k where j:k + 1 < length xs remdups-adj xs ! i =
xs ! k
        remdups-adj xs ! (i + 1) = xs ! (k + 1) by auto
    thus thesis using Cons(2) rem2 by auto
  next
    case False
    with rem xs have rem2:remdups-adj (x # xs) = x # remdups-adj xs by auto
    show thesis
```

proof (*cases i*)
case *0*
have *0 + 1 < length (x # xs)* **using** *lenxs* **by** *auto*
moreover have *remdups-adj (x # xs) ! i = (x # xs) ! 0*
proof −
have *remdups-adj (x # xs) ! i = (x # remdups-adj (y # xs′)) ! 0* **using** *xs rem2 0* **by** *simp*
also have *... = x* **by** *simp*
also have *... = (x # xs) ! 0* **by** *simp*
finally show *?thesis*.
qed
moreover have *remdups-adj (x # xs) ! (i + 1) = (x # xs) ! (0 + 1)*
proof −
have *remdups-adj (x # xs) ! (i + 1) = (x # remdups-adj (y # xs′)) ! 1* **using** *xs rem2 0* **by** *simp*
also have *... = remdups-adj (y # xs′) ! 0* **by** *simp*
also have *... = (y # (remdups (y # xs′))) ! 0* **by** (*metis nth-Cons′ remdups-adj-Cons-alt*)
also have *... = y* **by** *simp*
also have *... = (x # xs) ! (0 + 1)* **unfolding** *xs* **by** *simp*
finally show *?thesis*.
qed
ultimately show *thesis* **by** (*rule Cons.prems*(*1*))
next
case (*Suc k*)
with *Cons*(*3*) **have** *k + 1 < length (remdups-adj (x # xs)) − 1* **by** *auto*
also have *... ≤ length (remdups-adj xs) + 1 − 1* **by** (*metis One-nat-def le-refl list.size*(*4*) *rem2*)
also have *... = length (remdups-adj xs)* **by** *simp*
finally have *k + 1 < length (remdups-adj xs)*.
with *Cons.IH lenxs* **obtain** *j* **where** *j*:*j + 1 < length xs remdups-adj xs ! k = xs ! j*
remdups-adj xs ! (k + 1) = xs ! (j + 1) **by** *auto*
from *j*(*1*) **have** *Suc j + 1 < length (x # xs)* **by** *simp*
moreover have *remdups-adj (x # xs) ! i = (x # xs) ! (Suc j)*
proof −
have *remdups-adj (x # xs) ! i = (x # remdups-adj xs) ! i* **using** *rem2* **by** *simp*
also have *... = (remdups-adj xs) ! k* **using** *Suc* **by** *simp*
also have *... = xs ! j* **using** *j*(*2*) .
also have *... = (x # xs) ! (Suc j)* **by** *simp*
finally show *?thesis* .
qed
moreover have *remdups-adj (x # xs) ! (i + 1) = (x # xs) ! (Suc j + 1)*
proof −
have *remdups-adj (x # xs) ! (i + 1) = (x # remdups-adj xs) ! (i + 1)* **using** *rem2* **by** *simp*
also have *... = (remdups-adj xs) ! (k + 1)* **using** *Suc* **by** *simp*
also have *... = xs ! (j + 1)* **using** *j*(*3*).

also have ... = (*x* # *xs*) ! (*Suc j* + *1*) **by** *simp*
finally show *?thesis*.
qed
ultimately show *thesis* **by** (*rule Cons.prems*(*1*))
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*) ⟹ *remdups-adj xs* ! *i* ≠ *remdups-adj xs* ! *Suc i*
proof (*induction xs arbitrary*: *i rule*: *remdups-adj.induct*)
case (*3 x y xs i*)
thus *?case* **by** (*cases i*, *cases x* = *y*) (*simp*, *auto simp*: *hd-conv-nth*[*symmetric*])
qed *simp-all*

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

lemma (**in** *composition-series*) *intersect-normal*:
assumes *finite*:*finite* (*carrier G*)
assumes *KG*:*K* ◁ *G*
shows *composition-series* (*G*⦇*carrier* := *K*⦈) (*remdups-adj* (*map* (λ*H*. *K* ∩ *H*) 𝔊))
unfolding *composition-series-def composition-series-axioms-def normal-series-def normal-series-axioms-def*
apply (*auto simp only*: *conjI del*: *equalityI*)
proof −
show *group* (*G*⦇*carrier* := *K*⦈) **using** *KG normal-imp-subgroup subgroup-imp-group*
by *auto*
next
— Show, that removing adjacent duplicates doesn't result in an empty list.
assume *remdups-adj* (*map* ((∩) *K*) 𝔊) = []
hence *map* ((∩) *K*) 𝔊 = [] **by** (*metis remdups-adj-Nil-iff*)
hence 𝔊 = [] **by** (*metis Nil-is-map-conv*)
with *notempty* **show** *False*..
next
— Show, that the head of the reduced list is still the trivial group
have 𝔊 = {**1**} # *tl* 𝔊 **using** *notempty hd* **by** (*metis list.sel*(*1*,*3*) *neq-Nil-conv*)
hence *map* ((∩) *K*) 𝔊 = *map* ((∩) *K*) ({**1**} # *tl* 𝔊) **by** *simp*
hence *remdups-adj* (*map* ((∩) *K*) 𝔊) = *remdups-adj* ((*K* ∩ {**1**}) # (*map* ((∩) *K*) (*tl* 𝔊))) **by** *simp*
also have ... = (*K* ∩ {**1**}) # *tl* (*remdups-adj* ((*K* ∩ {**1**}) # (*map* ((∩) *K*) (*tl* 𝔊)))) **by** *simp*
finally have *hd* (*remdups-adj* (*map* ((∩) *K*) 𝔊)) = *K* ∩ {**1**} **using** *list.sel*(*1*)
by *metis*
thus *hd* (*remdups-adj* (*map* ((∩) *K*) 𝔊)) = {**1**$_{G⦇carrier := K⦈}$}
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* ((∩) *K*) $\mathfrak{G}$) = *map* ((∩) *K*) ((*carrier G*) # *tl* (*rev* $\mathfrak{G}$)) **by** (*metis rev-map*)
hence *rev*:*rev* (*map* ((∩) *K*) $\mathfrak{G}$) = (*K* ∩ (*carrier G*)) # (*map* ((∩) *K*) (*tl* (*rev* $\mathfrak{G}$))) **by** *simp*
have *last* (*remdups-adj* (*map* ((∩) *K*) $\mathfrak{G}$)) = *hd* (*rev* (*remdups-adj* (*map* ((∩) *K*) $\mathfrak{G}$)))
by (*metis hd-rev map-is-Nil-conv notempty remdups-adj-Nil-iff*)
also have ... = *hd* (*remdups-adj* (*rev* (*map* ((∩) *K*) $\mathfrak{G}$))) **by** (*metis remdups-adj-rev*)
also have ... = *hd* (*remdups-adj* ((*K* ∩ (*carrier G*)) # (*map* ((∩) *K*) (*tl* (*rev* $\mathfrak{G}$))))) **by** (*metis rev*)
also have ... = *hd* ((*K* ∩ (*carrier G*)) # (*remdups-adj* ((*K* ∩ (*carrier G*)) # (*map* ((∩) *K*) (*tl* (*rev* $\mathfrak{G}$)))))) **by** (*metis list.sel*(*1*) *remdups-adj-Cons-alt*)
also have ... = *K* **using** *KG normal-imp-subgroup subgroup.subset* **by** *force*
finally show *last* (*remdups-adj* (*map* ((∩) *K*) $\mathfrak{G}$)) = *carrier* (*G*⦇*carrier* := *K*⦈) **by** *auto*
next
— The induction step, using the second isomorphism theorem for groups.
fix *j*
assume *j*:*j* + *1* < *length* (*remdups-adj* (*map* ((∩) *K*) $\mathfrak{G}$))
have *KGnotempty*:(*map* ((∩) *K*) $\mathfrak{G}$) ≠ [] **using** *notempty* **by** (*metis Nil-is-map-conv*)
with *j* **obtain** *i* **where** *i*:*i* + *1* < *length* (*map* ((∩) *K*) $\mathfrak{G}$)
(*remdups-adj* (*map* ((∩) *K*) $\mathfrak{G}$)) ! *j* = (*map* ((∩) *K*) $\mathfrak{G}$) ! *i*
(*remdups-adj* (*map* ((∩) *K*) $\mathfrak{G}$)) ! (*j* + *1*) = (*map* ((∩) *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* ◁ *G*⦇*carrier* := $\mathfrak{G}$! (*i* + *1*)⦈ **by** (*metis normal*)
hence *GiSi'*:$\mathfrak{G}$! *i* ⊆ $\mathfrak{G}$! (*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** *GSiKnormGSi*:$\mathfrak{G}$! (*i* + *1*) ∩ *K* ◁ *G*⦇*carrier* := $\mathfrak{G}$! (*i* + *1*)⦈
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* ∩ ($\mathfrak{G}$! (*i* + *1*) ∩ *K*) ◁ *G*⦇*carrier* := $\mathfrak{G}$! (*i* + *1*)⦈
by (*metis group.normal-subgroup-intersect group.subgroup-imp-group i' is-group is-normal-series normal-series.normal-series-subgroups*)
hence *K* ∩ ($\mathfrak{G}$! *i* ∩ $\mathfrak{G}$! (*i* + *1*)) ◁ *G*⦇*carrier* := $\mathfrak{G}$! (*i* + *1*)⦈ **by** (*metis inf-commute inf-left-commute*)
hence *KGinormGSi*:*K* ∩ $\mathfrak{G}$! *i* ◁ *G*⦇*carrier* := $\mathfrak{G}$! (*i* + *1*)⦈ **using** *GiSi'* **by** (*metis le-iff-inf*)
moreover have *K* ∩ $\mathfrak{G}$! *i* ⊆ *K* ∩ $\mathfrak{G}$! (*i* + *1*) **using** *GiSi'* **by** *auto*
moreover have *groupGSi*:*group* (*G*⦇*carrier* := $\mathfrak{G}$! (*i* + *1*)⦈) **using** *i nor-*

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

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

with *i* **show** *simple-group* (*G*(|*carrier* := *remdups-adj* (*map* ((∩) *K*) 𝔊) ! *Suc j*|) *Mod remdups-adj* (*map* ((∩) *K*) 𝔊) ! *j*) **by** *auto*
qed
qed

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

lemma (**in** *composition-series*) *entries-mono*:
assumes *i* ≤ *j j* < *length* 𝔊
shows 𝔊 ! *i* ⊆ 𝔊 ! *j*
using *assms* **proof** (*induction j* − *i arbitrary*: *i j*)
case *0*
hence *i* = *j* **by** *auto*
thus 𝔊 ! *i* ⊆ 𝔊 ! *j* **by** *auto*
next
case (*Suc k i j*)
hence *i'*:*i* + (*Suc k*) = *j i* + *1* < *length* 𝔊 **by** *auto*
hence *ij*:*i* + *1* ≤ *j* **by** *auto*
have 𝔊 ! *i* ⊆ 𝔊 ! (*i* + *1*) **using** *i' normal normal-imp-subgroup subgroup.subset* **by** *force*

moreover have $j - (i + 1) = k$ $j < \mathit{length}\ \mathfrak{G}$ **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* = $\{G :: 'a\ \mathit{monoid}.\ \mathit{group}\ G\}$
proof
show $\bigwedge a.\ (\!|\mathit{carrier} = \{a\},\ \mathit{mult} = (\lambda x\ y.\ x),\ \mathit{one} = a|\!) \in \{G.\ \mathit{group}\ G\}$
unfolding *group-def group-axioms-def monoid-def Units-def* **by** *auto*
qed

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

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 φ **where** $\varphi \in \mathit{iso}\ (\mathit{Rep\text{-}group}\ G)\ (\mathit{Rep\text{-}group}\ H)$ **unfolding** *group-iso-rel-def* **by** *auto*
then obtain φ' **where** $\varphi' \in \mathit{iso}\ (\mathit{Rep\text{-}group}\ H)\ (\mathit{Rep\text{-}group}\ G)$ **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-iso-rel G H group-iso-rel H I*
then obtain φ ψ **where** $\varphi \in$ *iso* (*Rep-group G*) (*Rep-group H*) $\psi \in$ *iso* (*Rep-group H*) (*Rep-group I*)
unfolding *group-iso-rel-def* **by** *auto*
then obtain π **where** $\pi \in$ *iso* (*Rep-group G*) (*Rep-group I*)
using *iso-set-trans* **by** *blast*
thus *group-iso-rel G I* **unfolding** *group-iso-rel-def* **by** *auto*
qed
qed

This assigns to a given group the group isomorphism class

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

Two isomorphic groups do indeed have the same isomorphism class:

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

end

theory *JordanHolder*
imports
CompositionSeries
MaximalNormalSubgroups
HOL−Library.Multiset
GroupIsoClasses
begin

10 The Jordan-Hölder Theorem

locale *jordan-hoelder* = *group*
+ *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*(|*carrier* := $\mathfrak{G}$! (*length* $\mathfrak{G}$ − *1* − *1*)|)) (*take* (*length* $\mathfrak{G}$ − *1*) $\mathfrak{G}$)
proof (*rule nth-equalityI*)
define *n* **where** *n* = *length* $\mathfrak{G}$ − *1*
hence *n* = *length* (*take n* $\mathfrak{G}$) *n* > *0 n* < *length* $\mathfrak{G}$ **using** *assms notempty* **by** *auto*
interpret *normal*$\mathfrak{G}$*butlast*: *normal-series* (*G*(|*carrier* := $\mathfrak{G}$! (*n* − *1*)|)) *take n* $\mathfrak{G}$
using *normal-series-prefix-closed* ‹*n* > *0*› ‹*n* < *length* $\mathfrak{G}$› **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* ‹*n* = *length* (*take n* $\mathfrak{G}$)› *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 ∀ *i*<*length* (*butlast quotients*). *butlast quotients* ! *i* = *normal*$\mathfrak{G}$*butlast.quotients* ! *i*
proof *auto*
fix *i*
assume *i*:*i* < *length quotients* − *Suc 0*
hence *i*′:*i* < *length* $\mathfrak{G}$ − *1 i* < *n i* + *1* < *n* **unfolding** *n-def* **using** *quotients-length* **by** *auto*
from *i* **have** *butlast quotients* ! *i* = *quotients* ! *i* **by** (*metis One-nat-def length-butlast nth-butlast*)
also have ... = *G*(|*carrier* := $\mathfrak{G}$! (*i* + *1*)|) *Mod* $\mathfrak{G}$! *i* **unfolding** *quotients-def* **using** *i*′*(1)* **by** *auto*
also have ... = *G*(|*carrier* := (*take n* $\mathfrak{G}$) ! (*i* + *1*)|) *Mod* (*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* = *normal-series.quotients* (*G*(|*carrier* := $\mathfrak{G}$! (*n* − *Suc 0*)|)) (*take n* $\mathfrak{G}$) ! *i* **by** *auto*
qed
thus ⋀*i*. *i* < *length* (*butlast quotients*)
⟹ *butlast quotients* ! *i*
= *normal-series.quotients* (*G*(|*carrier* := $\mathfrak{G}$! (*length* $\mathfrak{G}$ − *1* − *1*)|)) (*take* (*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** *grpG*: *group G* **by** *simp*
  show *?case*
  proof (*cases length* $\mathfrak{G} \leq$ *2*)
  next
    case *True*
    hence *length* $\mathfrak{G}$ = *0* ∨ *length* $\mathfrak{G}$ = *1* ∨ *length* $\mathfrak{G}$ = *2* **by** *arith*
    with *comp*$\mathfrak{G}$.*notempty* **have** *length* $\mathfrak{G}$ = *1* ∨ *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 *length*:*length* $\mathfrak{G}$ ≥ *3* **by** *simp*
    — First we show that $\mathfrak{H}$ must have a length of at least 3.

hence ¬ *simple-group G* **using** *comp𝔊.composition-series-simple-group* **by** *auto*
hence $\mathfrak{H} \neq [\{\mathbf{1}_G\}, \textit{carrier } G]$ **using** *comp𝔊.composition-series-simple-group* **by** *auto*
hence *length* $\mathfrak{H} \neq 2$ **using** *comp𝔊.length-two-unique* **by** *auto*
moreover from *length* **have** *carrier* $G \neq \{\mathbf{1}_G\}$ **using** *comp𝔊.composition-series-length-one comp𝔊.composition-series-triv-group* **by** *auto*
hence *length* $\mathfrak{H} \neq 1$ **using** *comp𝔊.composition-series-length-one comp𝔊.composition-series-triv-group* **by** *auto*
moreover from *comp𝔊.notempty* **have** *length* $\mathfrak{H} \neq 0$ **by** *simp*
ultimately have *length𝔊big*:*length* $\mathfrak{H} \geq 3$ **using** *comp𝔊.notempty* **by** *arith*
define *m* **where** $m = \textit{length } \mathfrak{H} - 1$
define *n* **where** $n = \textit{length } \mathfrak{G} - 1$
from *length𝔊big* **have** *m'*:$m > 0$ $m < \textit{length } \mathfrak{H}$ $(m - 1) + 1 < \textit{length } \mathfrak{H}$ $m - 1 = \textit{length } \mathfrak{H} - 2$ $m - 1 + 1 = \textit{length } \mathfrak{H} - 1$ $m - 1 < \textit{length } \mathfrak{H}$
unfolding *m-def* **by** *auto*
from *length* **have** *n'*:$n > 0$ $n < \textit{length } \mathfrak{G}$ $(n - 1) + 1 < \textit{length } \mathfrak{G}$ $n - 1 < \textit{length } \mathfrak{G}$ *Suc n* $\leq$ *length* $\mathfrak{G}$ $n - 1 = \textit{length } \mathfrak{G} - 2$ $n - 1 + 1 = \textit{length } \mathfrak{G} - 1$ **unfolding** *n-def* **by** *auto*
define $\mathfrak{G}Pn$ **where** $\mathfrak{G}Pn = G(\!|\textit{carrier} := \mathfrak{G} ! (n - 1)|\!)$
define $\mathfrak{H}Pm$ **where** $\mathfrak{H}Pm = G(\!|\textit{carrier} := \mathfrak{H} ! (m - 1)|\!)$
then interpret *grp𝔊Pn*: *group* $\mathfrak{G}Pn$ **unfolding** $\mathfrak{G}Pn$*-def* **using** *n'* **by** (*metis comp𝔊.normal-series-subgroups comp𝔊.subgroup-imp-group*)
interpret *grp𝔊Pm*: *group* $\mathfrak{H}Pm$ **unfolding** $\mathfrak{H}Pm$*-def* **using** *m' comp𝔊.normal-series-subgroups 1(2) group.subgroup-imp-group* **by** *force*
have *finGbl*:*finite* (*carrier* $\mathfrak{G}Pn$) **using** ‹$n - 1 < \textit{length } \mathfrak{G}$› *1(3)* **unfolding** $\mathfrak{G}Pn$*-def* **using** *comp𝔊.normal-series-subgroups comp𝔊.subgroup-finite* **by** *auto*
have *finHbl*:*finite* (*carrier* $\mathfrak{H}Pm$) **using** ‹$m - 1 < \textit{length } \mathfrak{H}$› *1(3)* **unfolding** $\mathfrak{H}Pm$*-def* **using** *comp𝔊.normal-series-subgroups comp𝔊.subgroup-finite* **by** *auto*
have *quots𝔊notempty*:*comp𝔊.quotients* $\neq$ [] **using** *comp𝔊.quotients-length length* **by** *auto*
have *quots𝔊notempty*:*comp𝔊.quotients* $\neq$ [] **using** *comp𝔊.quotients-length length𝔊big* **by** *auto*

— Instantiate truncated composition series since they are used for both cases
interpret $\mathfrak{H}$*butlast*: *composition-series* $\mathfrak{H}Pm$ *take m* $\mathfrak{H}$ **using** *comp𝔊.composition-series-prefix-closed m'(1,2)* $\mathfrak{H}Pm$*-def* **by** *auto*
interpret $\mathfrak{G}$*butlast*: *composition-series* $\mathfrak{G}Pn$ *take n* $\mathfrak{G}$ **using** *comp𝔊.composition-series-prefix-closed n'(1,2)* $\mathfrak{G}Pn$*-def* **by** *auto*
have *ltaken*:n = *length* (*take n* $\mathfrak{G}$) **using** *length-take n'(2)* **by** *auto*
have *ltakem*:m = *length* (*take m* $\mathfrak{H}$) **using** *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 *lasteq*:*last comp𝔊.quotients* = *last comp𝔊.quotients*
proof −
from *length* **have** *lg*:*length* $\mathfrak{G} - 1 - 1 + 1 = \textit{length } \mathfrak{G} - 1$ **by** (*metis Suc-diff-1 Suc-eq-plus1 n'(1) n-def*)

from *length*$\mathfrak{H}$*big* **have** *lh*:*length* $\mathfrak{H}$ − *1* − *1* + *1* = *length* $\mathfrak{H}$ − *1* **by** (*metis Suc-diff-1 Suc-eq-plus1* ‹*0* < *m*› *m-def*)
have *last comp*$\mathfrak{G}$*.quotients* = *G Mod* $\mathfrak{G}$! (*n* − *1*) **using** *length comp*$\mathfrak{G}$*.last-quotient* **unfolding** *n-def* **by** *auto*
also have ... = *G Mod* $\mathfrak{H}$! (*m* − *1*) **using** *True* **by** *simp*
also have ... = *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* (*map group.iso-class* $\mathfrak{G}$*butlast.quotients*) = *mset* (*map group.iso-class* $\mathfrak{H}$*butlast.quotients*)
using *1*(*1*) *True n'*(*5*) *grp*$\mathfrak{G}$*Pn.is-group finGbl* $\mathfrak{G}$*butlast.is-composition-series* $\mathfrak{H}$*butlast.is-composition-series* **unfolding** $\mathfrak{G}$*Pn-def* $\mathfrak{H}$*Pm-def* **by** *metis*
have *mset* (*map group.iso-class comp*$\mathfrak{G}$*.quotients*)
= *mset* (*map group.iso-class* (*butlast comp*$\mathfrak{G}$*.quotients* @ [*last comp*$\mathfrak{G}$*.quotients*])) **by** (*simp add*: *quots*$\mathfrak{G}$*notempty*)
also have ... = *mset* (*map group.iso-class* ($\mathfrak{G}$*butlast.quotients* @ [*last* (*comp*$\mathfrak{G}$*.quotients*)])) **using** *comp*$\mathfrak{G}$*.quotients-butlast length* **unfolding** *n-def* $\mathfrak{G}$*Pn-def* **by** *auto*
also have ... = *mset* ((*map group.iso-class* $\mathfrak{G}$*butlast.quotients*) @ [*group.iso-class* (*last* (*comp*$\mathfrak{G}$*.quotients*))]) **by** *auto*
also have ... = *mset* (*map group.iso-class* $\mathfrak{G}$*butlast.quotients*) + {# *group.iso-class* (*last* (*comp*$\mathfrak{G}$*.quotients*)) #} **by** *auto*
also have ... = *mset* (*map group.iso-class* $\mathfrak{H}$*butlast.quotients*) + {# *group.iso-class* (*last* (*comp*$\mathfrak{G}$*.quotients*)) #} **using** *ind* **by** *simp*
also have ... = *mset* (*map group.iso-class* $\mathfrak{H}$*butlast.quotients*) + {# *group.iso-class* (*last* (*comp*$\mathfrak{H}$*.quotients*)) #} **using** *lasteq* **by** *simp*
also have ... = *mset* ((*map group.iso-class* $\mathfrak{H}$*butlast.quotients*) @ [*group.iso-class* (*last* (*comp*$\mathfrak{H}$*.quotients*))]) **by** *auto*
also have ... = *mset* (*map group.iso-class* ($\mathfrak{H}$*butlast.quotients* @ [*last* (*comp*$\mathfrak{H}$*.quotients*)])) **by** *auto*
also have ... = *mset* (*map group.iso-class* (*butlast comp*$\mathfrak{H}$*.quotients* @ [*last comp*$\mathfrak{H}$*.quotients*])) **using** *length*$\mathfrak{H}$*big comp*$\mathfrak{H}$*.quotients-butlast* **unfolding** *m-def* $\mathfrak{H}$*Pm-def* **by** *auto*
also have ... = *mset* (*map group.iso-class comp*$\mathfrak{H}$*.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*(| *carrier* := $\mathfrak{H}$! (*m* − *1*) ∩ $\mathfrak{G}$! (*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*) ◁ *G* **using** *comp*$\mathfrak{H}$*.normal-series-snd-to-last m'*(*4*) **unfolding** *m-def* **by** *auto*

have $\mathfrak{G}Pnnorm G$:$\mathfrak{G}$! $(n - 1) \lhd G$ **using** *comp*$\mathfrak{G}$*.normal-series-snd-to-last* $n'(6)$ **unfolding** *n-def* **by** *auto*

have $\mathfrak{H}Pmint\mathfrak{G}Pnnorm G$:$\mathfrak{H}$! $(m - 1) \cap \mathfrak{G}$! $(n - 1) \lhd G$ **using** $\mathfrak{H}Pmnorm G$ $\mathfrak{G}Pnnorm G$ **by** (*rule comp*$\mathfrak{G}$*.normal-subgroup-intersect*)

have *Intnorm*$\mathfrak{G}Pn$:$\mathfrak{H}$! $(m - 1) \cap \mathfrak{G}$! $(n - 1) \lhd \mathfrak{G}Pn$ **using** $\mathfrak{G}Pnnorm G$ $\mathfrak{H}Pmnorm G$ *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*$\mathfrak{H}Pm$:$\mathfrak{H}$! $(m - 1) \cap \mathfrak{G}$! $(n - 1) \lhd \mathfrak{H}Pm$ **using** $\mathfrak{H}Pmnorm G$ $\mathfrak{G}Pnnorm G$ *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** $\mathfrak{H}Pmmax.max\text{-}normal$ $\mathfrak{G}Pnnorm G$ *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** $\mathfrak{G}Pnmax.max\text{-}normal$ $\mathfrak{H}Pmnorm G$ *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 $\mathfrak{H}PmSubSetmult$:$\mathfrak{H}$! $(m - 1) \subseteq \mathfrak{H}$! $(m - 1) <\#>_G \mathfrak{G}$! $(n - 1)$

using *second-isomorphism-grp.H-contained-in-set-mult* $\mathfrak{G}Pnmax.is\text{-}normal$ $\mathfrak{H}Pmnorm G$ *normal-imp-subgroup*

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

have $\mathfrak{G}PnSubSetmult$:$\mathfrak{G}$! $(n - 1) \subseteq \mathfrak{H}$! $(m - 1) <\#>_G \mathfrak{G}$! $(n - 1)$

using *second-isomorphism-grp.S-contained-in-set-mult* $\mathfrak{G}Pnmax.is\text{-}normal$ $\mathfrak{H}Pmnorm G$ *normal-imp-subgroup*

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

have $\mathfrak{G}$! $(n - 1) \neq (\mathfrak{H}$! $(m - 1)) <\#>_G (\mathfrak{G}$! $(n - 1))$ **using** $\mathfrak{H}PmSubSetmult$ *not*$\mathfrak{H}PmSub\mathfrak{G}Pn$ **by** *auto*

hence *set-mult G*:$(\mathfrak{H}$! $(m - 1)) <\#>_G (\mathfrak{G}$! $(n - 1)) =$ *carrier G*

using $\mathfrak{G}Pnmax.max\text{-}normal$ $\mathfrak{G}Pnmax.is\text{-}normal$ $\mathfrak{H}Pmnorm G$ *comp*$\mathfrak{G}$*.normal-subgroup-set-mult-closed* $\mathfrak{G}PnSubSetmult$ **by** *metis*

then obtain φ **where** $\varphi \in$ *iso* $(\mathfrak{G}Pn$ *Mod* $(\mathfrak{H}$! $(m - 1) \cap \mathfrak{G}$! $(n - 1)))$ $(G(\|carrier := carrier\ G\|)$ *Mod* $\mathfrak{H}$! $(m - 1))$

using *second-isomorphism-grp.normal-intersection-quotient-isom* $\mathfrak{H}Pmnorm G$ $\mathfrak{G}Pnmax.is\text{-}normal$ *normal-imp-subgroup*

unfolding *second-isomorphism-grp-def second-isomorphism-grp-axioms-def max-normal-subgroup-def* $\mathfrak{G}Pn$*-def* **by** *metis*

hence φ:$\varphi \in$ *iso* $(\mathfrak{G}Pn$ *Mod* $(\mathfrak{H}$! $(m - 1) \cap \mathfrak{G}$! $(n - 1)))$ $(G$ *Mod* $\mathfrak{H}$! $(m - 1))$ **by** *auto*

then obtain $\varphi 2$ **where** $\varphi 2$:$\varphi 2 \in$ *iso* $(G$ *Mod* $\mathfrak{H}$! $(m - 1))$ $(\mathfrak{G}Pn$ *Mod* $(\mathfrak{H}$! $(m - 1) \cap \mathfrak{G}$! $(n - 1)))$

using *group.iso-set-sym grp𝔊PnMod𝔎Pmint𝔊Pn.is-group* **by** *auto*
moreover have *simple-group* (*G*⦇*carrier* := 𝔎 ! (*m* − *1* + *1*)⦈ *Mod* 𝔎 ! (*m* − *1*)) **using** *comp𝔎.simplefact m'(3)* **by** *simp*
hence *simple-group* (*G Mod* 𝔎 ! (*m* − *1*)) **using** *comp𝔎.last last-conv-nth comp𝔎.notempty m'(5)* **by** *fastforce*
ultimately have *simple𝔊PnModInt*:*simple-group* (𝔊*Pn Mod* (𝔎 ! (*m* − *1*) ∩ 𝔊 ! (*n* − *1*)))
using *simple-group.iso-simple grp𝔊PnMod𝔎Pmint𝔊Pn.is-group* **by** *auto*
interpret *grpGMod𝔎Pm*: *group* (*G Mod* 𝔎 ! (*m* − *1*)) **by** (*metis 𝔎Pmnorm G normal.factorgroup-is-group*)

— Show analogues of the previous statements for 𝔎 ! (*m* − *1*) instead of 𝔊 ! (*n* − *1*).
have *𝔎PmSubSetmult'*:𝔎 ! (*m* − *1*) ⊆ 𝔊 ! (*n* − *1*) <#>$_G$ 𝔎 ! (*m* − *1*)
using *second-isomorphism-grp.S-contained-in-set-mult 𝔊Pnmax.is-normal 𝔎PmnormG normal-imp-subgroup*
unfolding *second-isomorphism-grp-def second-isomorphism-grp-axioms-def max-normal-subgroup-def* **by** *metis*
have *𝔊PnSubSetmult'*:𝔊 ! (*n* − *1*) ⊆ 𝔊 ! (*n* − *1*) <#>$_G$ 𝔎 ! (*m* − *1*)
using *second-isomorphism-grp.H-contained-in-set-mult 𝔊Pnmax.is-normal 𝔎PmnormG normal-imp-subgroup*
unfolding *second-isomorphism-grp-def second-isomorphism-grp-axioms-def max-normal-subgroup-def* **by** *metis*
have 𝔎 ! (*m* − *1*) ≠ (𝔊 ! (*n* − *1*)) <#>$_G$ (𝔎 ! (*m* − *1*)) **using** *𝔊PnSubSetmult' not𝔊PnSub𝔎Pm* **by** *auto*
hence *set-multG*:(𝔊 ! (*n* − *1*)) <#>$_G$ (𝔎 ! (*m* − *1*)) = *carrier G*
using *𝔎Pmmax.max-normal 𝔎Pmmax.is-normal 𝔊PnnormG comp𝔊.normal-subgroup-set-mult-closed 𝔎PmSubSetmult'* **by** *metis*
from *set-multG* **obtain** *ψ* **where**
ψ ∈ *iso* (𝔎*Pm Mod* (𝔊 ! (*n* − *1*) ∩ 𝔎 ! (*m* − *1*))) (*G*⦇*carrier* := *carrier G*⦈ *Mod* 𝔊 ! (*n* − *1*))
using *second-isomorphism-grp.normal-intersection-quotient-isom 𝔊PnnormG 𝔎Pmmax.is-normal normal-imp-subgroup*
unfolding *second-isomorphism-grp-def second-isomorphism-grp-axioms-def max-normal-subgroup-def 𝔎Pm-def* **by** *metis*
hence *ψ*:*ψ* ∈ *iso* (𝔎*Pm Mod* (𝔎 ! (*m* − *1*) ∩ (𝔊 ! (*n* − *1*)))) (*G*⦇*carrier* := *carrier G*⦈ *Mod* 𝔊 ! (*n* − *1*)) **using** *Int-commute* **by** *metis*
then obtain *ψ2* **where**
ψ2:*ψ2* ∈ *iso* (*G Mod* 𝔊 ! (*n* − *1*)) (𝔎*Pm Mod* (𝔎 ! (*m* − *1*) ∩ 𝔊 ! (*n* − *1*)))
using *group.iso-set-sym grp𝔎PmMod𝔎Pmint𝔊Pn.is-group* **by** *auto*
moreover have *simple-group* (*G*⦇*carrier* := 𝔊 ! (*n* − *1* + *1*)⦈ *Mod* 𝔊 ! (*n* − *1*)) **using** *comp𝔊.simplefact n'(3)* **by** *simp*
hence *simple-group* (*G Mod* 𝔊 ! (*n* − *1*)) **using** *comp𝔊.last last-conv-nth comp𝔊.notempty n'(7)* **by** *fastforce*
ultimately have *simple𝔎PmModInt*:*simple-group* (𝔎*Pm Mod* (𝔎 ! (*m* − *1*) ∩ 𝔊 ! (*n* − *1*)))
using *simple-group.iso-simple grp𝔎PmMod𝔎Pmint𝔊Pn.is-group* **by** *auto*
interpret *grpGMod𝔊Pn*: *group* (*G Mod* 𝔊 ! (*n* − *1*)) **by** (*metis 𝔊PnnormG*

normal.factorgroup-is-group)

— Instantiate several composition series used to build up the equality of quotient multisets.
define $\mathfrak{K}$ **where** $\mathfrak{K}$ = *remdups-adj* (*map* ((∩) ($\mathfrak{H}$! (*m* − *1*))) $\mathfrak{G}$)
define $\mathfrak{L}$ **where** $\mathfrak{L}$ = *remdups-adj* (*map* ((∩) ($\mathfrak{G}$! (*n* − *1*))) $\mathfrak{H}$)
interpret $\mathfrak{K}$: *composition-series* $\mathfrak{H}$*Pm* $\mathfrak{K}$ **using** *comp*$\mathfrak{G}$.*intersect-normal 1*(*3*) $\mathfrak{H}$*PmnormG* **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* $\mathfrak{G}$)) ≤ *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* ≤ *length* $\mathfrak{L}$ **using** *n*′(*6*) *length* **by** *auto*
have *Inteq*$\mathfrak{L}$*sndlast*:$\mathfrak{H}$! (*m* − *1*) ∩ $\mathfrak{G}$! (*n* − *1*) = $\mathfrak{L}$! (*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* ((∩) ($\mathfrak{G}$! (*n* − *1*))) $\mathfrak{H}$) ≠ [] **using** *comp*$\mathfrak{H}$.*notempty* **by** (*metis Nil-is-map-conv*)
ultimately obtain *i* **where** *i*:*i* + *1* < *length* (*map* ((∩) ($\mathfrak{G}$! (*n* − *1*))) $\mathfrak{H}$) $\mathfrak{L}$! (*length* $\mathfrak{L}$ − *1* − *1*) = (*map* ((∩) ($\mathfrak{G}$! (*n* − *1*))) $\mathfrak{H}$) ! *i* $\mathfrak{L}$! (*length* $\mathfrak{L}$ − *1* − *1* + *1*) = (*map* ((∩) ($\mathfrak{G}$! (*n* − *1*))) $\mathfrak{H}$) ! (*i* + *1*)
using *remdups-adj-obtain-adjacency* **unfolding** $\mathfrak{L}$*-def* **by** *force*
hence $\mathfrak{L}$! (*length* $\mathfrak{L}$ − *1* − *1*) = $\mathfrak{H}$! *i* ∩ $\mathfrak{G}$! (*n* − *1*) $\mathfrak{L}$! (*length* $\mathfrak{L}$ − *1* − *1* + *1*) = $\mathfrak{H}$! (*i* + *1*) ∩ $\mathfrak{G}$! (*n* − *1*) **by** *auto*
hence $\mathfrak{L}$! (*length* $\mathfrak{L}$ − *1*) = $\mathfrak{H}$! (*i* + *1*) ∩ $\mathfrak{G}$! (*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*) ⊆ $\mathfrak{H}$! (*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 ¬ (*i* + *1* ≤ *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*(|*carrier* := $\mathfrak{L}$! (*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 ‹*length* $\mathfrak{L}$ > *1*› **have** *quots*$\mathfrak{L}$*notemtpy*:$\mathfrak{L}$*.quotients* ≠ [] **unfolding** $\mathfrak{L}$*.quotients-def* **by** *auto*

— Apply the induction hypothesis on $\mathfrak{L}butlast$ and $\mathfrak{K}butlast$
have *length* $\mathfrak{K} > 1$
proof (*rule ccontr*)
assume ¬ *length* $\mathfrak{K} > 1$
with $\mathfrak{K}.notempty$ **have** *length* $\mathfrak{K} = 1$ **by** (*metis One-nat-def Suc-lessI length-greater-0-conv*)
hence *carrier* $\mathfrak{H}Pm = \{\mathbf{1}_{\mathfrak{H}Pm}\}$ **using** $\mathfrak{K}.composition\text{-}series\text{-}length\text{-}one$ $\mathfrak{K}.composition\text{-}series\text{-}triv\text{-}group$ **by** *auto*
hence *carrier* $\mathfrak{H}Pm = \{\mathbf{1}_G\}$ **unfolding** $\mathfrak{H}Pm\text{-}def$ **by** *auto*
hence *carrier* $\mathfrak{H}Pm \subseteq \mathfrak{G}\ !\ (n - 1)$ **using** $\mathfrak{G}Pnmax.is\text{-}subgroup$ *subgroup.one-closed* **by** *auto*
with $not\mathfrak{H}Pm Sub\mathfrak{G}Pn$ **show** *False* **unfolding** $\mathfrak{H}Pm\text{-}def$ **by** *auto*
qed
hence $length\mathfrak{K}'$:*length* $\mathfrak{K} - 1 > 0$ *length* $\mathfrak{K} - 1 \leq$ *length* $\mathfrak{K}$ **by** *auto*
have $Inteq\mathfrak{K}sndlast$:$\mathfrak{H}\ !\ (m - 1) \cap \mathfrak{G}\ !\ (n - 1) = \mathfrak{K}\ !\ (length\ \mathfrak{K} - 1 - 1)$
proof −
have *length* $\mathfrak{K} - 1 - 1 + 1 <$ *length* $\mathfrak{K}$ **using** $length\mathfrak{K}'$ **by** *auto*
moreover have $KGnotempty$:(*map* ((∩) ($\mathfrak{H}\ !\ (m - 1)$)) $\mathfrak{G}$) ≠ [] **using** $comp\mathfrak{G}.notempty$ **by** (*metis Nil-is-map-conv*)
ultimately obtain *i* **where** *i*:*i* + 1 < *length* (*map* ((∩) ($\mathfrak{H}\ !\ (m - 1)$)) $\mathfrak{G}$) $\mathfrak{K}\ !\ (length\ \mathfrak{K} - 1 - 1) = (map\ ((\cap)\ (\mathfrak{H}\ !\ (m - 1)))\ \mathfrak{G})\ !\ i\ \mathfrak{K}\ !\ (length\ \mathfrak{K} - 1 - 1 + 1) = (map\ ((\cap)\ (\mathfrak{H}\ !\ (m - 1)))\ \mathfrak{G})\ !\ (i + 1)$
using *remdups-adj-obtain-adjacency* **unfolding** $\mathfrak{K}\text{-}def$ **by** *force*
hence $\mathfrak{K}\ !\ (length\ \mathfrak{K} - 1 - 1) = \mathfrak{G}\ !\ i \cap \mathfrak{H}\ !\ (m - 1)\ \mathfrak{K}\ !\ (length\ \mathfrak{K} - 1 - 1 + 1) = \mathfrak{G}\ !\ (i + 1) \cap \mathfrak{H}\ !\ (m - 1)$ **by** *auto*
hence $\mathfrak{K}\ !\ (length\ \mathfrak{K} - 1) = \mathfrak{G}\ !\ (i + 1) \cap \mathfrak{H}\ !\ (m - 1)$ **using** $length\mathfrak{K}'(1)$ **by** (*metis Suc-diff-1 Suc-eq-plus1*)
hence $\mathfrak{H}Pmsub\mathfrak{G}Pn$:$\mathfrak{H}\ !\ (m - 1) \subseteq \mathfrak{G}\ !\ (i + 1)$ **using** $\mathfrak{K}.last$ $\mathfrak{K}.notempty$ *last-conv-nth* **unfolding** $\mathfrak{H}Pm\text{-}def$ **by** *auto*
from *i*(*1*) **have** *i* + 1 < *n* + 1 **unfolding** *n-def* **by** *auto*
moreover have ¬ (*i* + 1 ≤ *n* − 1) **using** $comp\mathfrak{G}.entries\text{-}mono$ $n'(2)$ $not\mathfrak{H}Pm Sub\mathfrak{G}Pn$ $\mathfrak{H}Pmsub\mathfrak{G}Pn$ **by** *fastforce*
ultimately have *n* − 1 = *i* **by** *auto*
with *i* **show** *?thesis* **by** *auto*
qed
have *composition-series* (G(|*carrier* := $\mathfrak{K}\ !\ (length\ \mathfrak{K} - 1 - 1)$|)) (*take* (*length* $\mathfrak{K} - 1$) $\mathfrak{K}$)
using $length\mathfrak{K}'$ $\mathfrak{K}.composition\text{-}series\text{-}prefix\text{-}closed$ **unfolding** $\mathfrak{H}PmInt\mathfrak{G}Pn\text{-}def$ $\mathfrak{H}Pm\text{-}def$ **by** *fastforce*
then interpret $\mathfrak{K}butlast$: *composition-series* $\mathfrak{H}PmInt\mathfrak{G}Pn$ (*take* (*length* $\mathfrak{K}$ − 1) $\mathfrak{K}$) **using** $Inteq\mathfrak{K}sndlast$ **unfolding** $\mathfrak{H}PmInt\mathfrak{G}Pn\text{-}def$ **by** *auto*
from *finGbl* **have** *finInt*:*finite* (*carrier* $\mathfrak{H}PmInt\mathfrak{G}Pn$) **unfolding** $\mathfrak{H}PmInt\mathfrak{G}Pn\text{-}def$ $\mathfrak{G}Pn\text{-}def$ **by** *simp*
moreover have *Suc* (*length* (*take* (*length* $\mathfrak{L}$ − 1) $\mathfrak{L}$)) ≤ *length* $\mathfrak{G}$ **using** $length\mathfrak{L}$ **unfolding** *n-def* **using** $n'(2)$ **by** *auto*
ultimately have $multisets\mathfrak{K}\mathfrak{L}butlast$:*mset* (*map group.iso-class* $\mathfrak{L}butlast.quotients$) = *mset* (*map group.iso-class* $\mathfrak{K}butlast.quotients$)
using *1.hyps* $\mathfrak{L}butlast.is\text{-}group$ $\mathfrak{K}butlast.is\text{-}composition\text{-}series$ $\mathfrak{L}butlast.is\text{-}composition\text{-}series$

by *auto*
hence *length* (*take* (*length* $\mathfrak{K}$ − *1*) $\mathfrak{K}$) = *length* (*take* (*length* $\mathfrak{L}$ − *1*) $\mathfrak{L}$)
using $\mathfrak{K}$*butlast.quotients-length* $\mathfrak{L}$*butlast.quotients-length length-map size-mset*
by *metis*
hence *length* (*take* (*length* $\mathfrak{K}$ − *1*) $\mathfrak{K}$) = *n* − *1* **using** *length*$\mathfrak{L}$ *n'*(*1*) **by** *auto*
hence *length*$\mathfrak{K}$:*length* $\mathfrak{K}$ = *n* **by** (*metis Suc-diff-1* $\mathfrak{K}$*.notempty butlast-conv-take*
length-butlast length-greater-0-conv n'(*1*))

— Apply the induction hypothesis on $\mathfrak{K}$ and $\mathfrak{H}$*butlast*
from *Inteq*$\mathfrak{K}$*sndlast* **have** $\mathfrak{K}$*sndlast*:$\mathfrak{H}$*PmInt*$\mathfrak{G}$*Pn* = ($\mathfrak{H}$*Pm*(|*carrier* := $\mathfrak{K}$! (*length* $\mathfrak{K}$ − *1* − *1*)|)) **unfolding** $\mathfrak{H}$*PmInt*$\mathfrak{G}$*Pn-def* $\mathfrak{H}$*Pm-def* $\mathfrak{K}$*-def* **by** *auto*
from *length*$\mathfrak{K}$ **have** *Suc* (*length* $\mathfrak{K}$) ≤ *length* $\mathfrak{G}$ **using** *n'*(*2*) **by** *auto*
hence *multisets*$\mathfrak{H}$*butlast*$\mathfrak{K}$:*mset* (*map group.iso-class* $\mathfrak{H}$*butlast.quotients*) =
mset (*map group.iso-class* $\mathfrak{K}$*.quotients*)
using *1.hyps grp*$\mathfrak{H}$*Pm.is-group finHbl* $\mathfrak{H}$*butlast.is-composition-series* $\mathfrak{K}$*.is-composition-series*
by *metis*
hence *length*$\mathfrak{K}$:*m* = *length* $\mathfrak{K}$ **using** $\mathfrak{H}$*butlast.quotients-length* $\mathfrak{K}$*.quotients-length*
length-map size-mset ltakem **by** *metis*
hence *length* $\mathfrak{K}$ > *1 length* $\mathfrak{K}$ − *1* > *0 length* $\mathfrak{K}$ − *1* ≤ *length* $\mathfrak{K}$ **using** *m'*(*4*)
length$\mathfrak{H}$*big* **by** *auto*
hence *quots*$\mathfrak{K}$*notemtpy*:$\mathfrak{K}$*.quotients* ≠ [] **unfolding** $\mathfrak{K}$*.quotients-def* **by** *auto*

interpret $\mathfrak{K}$*butlastadd*$\mathfrak{G}$*Pn*: *composition-series* $\mathfrak{G}$*Pn* (*take* (*length* $\mathfrak{K}$ − *1*) $\mathfrak{K}$)
@ [$\mathfrak{G}$! (*n* − *1*)]
using *grp*$\mathfrak{G}$*Pn.composition-series-extend* $\mathfrak{K}$*butlast.is-composition-series sim-*
ple$\mathfrak{G}$*PnModInt Intnorm*$\mathfrak{G}$*Pn*
unfolding $\mathfrak{G}$*Pn-def* $\mathfrak{H}$*PmInt*$\mathfrak{G}$*Pn-def* **by** *auto*
interpret $\mathfrak{L}$*butlastadd*$\mathfrak{H}$*Pm*: *composition-series* $\mathfrak{H}$*Pm* (*take* (*length* $\mathfrak{L}$ − *1*) $\mathfrak{L}$)
@ [$\mathfrak{H}$! (*m* − *1*)]
using *grp*$\mathfrak{H}$*Pm.composition-series-extend* $\mathfrak{L}$*butlast.is-composition-series*
simple$\mathfrak{H}$*PmModInt Intnorm*$\mathfrak{H}$*Pm*
unfolding $\mathfrak{H}$*Pm-def* $\mathfrak{H}$*PmInt*$\mathfrak{G}$*Pn-def* **by** *auto*

— Prove equality of those composition series.
have *mset* (*map group.iso-class comp*$\mathfrak{G}$*.quotients*)
= *mset* (*map group.iso-class* ((*butlast comp*$\mathfrak{G}$*.quotients*) @ [*last*
comp$\mathfrak{G}$*.quotients*])) **using** *quots*$\mathfrak{G}$*notempty* **by** *simp*
also have ... = *mset* (*map group.iso-class* ($\mathfrak{G}$*butlast.quotients* @ [*G Mod* $\mathfrak{G}$!
(*n* − *1*)]))
using *comp*$\mathfrak{G}$*.quotients-butlast comp*$\mathfrak{G}$*.last-quotient length* **unfolding** *n-def*
$\mathfrak{G}$*Pn-def* **by** *auto*
also have ... = *mset* (*map group.iso-class* ((*butlast* $\mathfrak{L}$*.quotients*) @ [*last*
$\mathfrak{L}$*.quotients*])) + {# *group.iso-class* (*G Mod* $\mathfrak{G}$! (*n* − *1*)) #}
using *multisets*$\mathfrak{G}$*butlast*$\mathfrak{L}$ *quots*$\mathfrak{L}$*notemtpy* **by** *simp*
also have ... = *mset* (*map group.iso-class* ($\mathfrak{L}$*butlast.quotients* @ [$\mathfrak{G}$*Pn Mod*
$\mathfrak{H}$! (*m* − *1*) ∩ $\mathfrak{G}$! (*n* − *1*)])) + {# *group.iso-class* (*G Mod* $\mathfrak{G}$! (*n* − *1*)) #}
using $\mathfrak{L}$*.quotients-butlast* $\mathfrak{L}$*.last-quotient* ‹*length* $\mathfrak{L}$ > *1*› $\mathfrak{L}$*sndlast Inteq*$\mathfrak{L}$*sndlast*
unfolding *n-def* **by** *auto*
also have ... = *mset* (*map group.iso-class* $\mathfrak{K}$*butlast.quotients*) + {# *group.iso-class*

```
(𝔊Pn Mod 𝔥 ! (m − 1) ∩ 𝔊 ! (n − 1)) #} + {# group.iso-class (G Mod 𝔊 ! (n
− 1)) #}
      using multisets𝔎𝔏butlast by simp
   also have ... = mset (map group.iso-class 𝔎butlast.quotients) + {# group.iso-class
(G Mod 𝔥 ! (m − 1)) #} + {# group.iso-class (𝔥Pm Mod 𝔥 ! (m − 1) ∩ 𝔊 ! (n
− 1)) #}
    using φ ψ2 iso-classes-iff grp𝔊PnMod𝔥Pmint𝔊Pn.is-group grpGMod𝔥Pm.is-group
grpGMod𝔊Pn.is-group grp𝔥PmMod𝔥Pmint𝔊Pn.is-group
      by metis
   also have ... = mset (map group.iso-class 𝔎butlast.quotients) + {# group.iso-class
(𝔥Pm Mod 𝔥 ! (m − 1) ∩ 𝔊 ! (n − 1)) #} + {# group.iso-class (G Mod 𝔥 ! (m
− 1)) #}
      by simp
      also have ... = mset (map group.iso-class ((butlast 𝔎.quotients) @ [last
𝔎.quotients])) + {# group.iso-class (G Mod 𝔥 ! (m − 1)) #}
    using 𝔎.quotients-butlast 𝔎.last-quotient ‹length 𝔎 > 1› 𝔎sndlast Inteq𝔎sndlast
unfolding m-def by auto
   also have ... = mset (map group.iso-class 𝔥butlast.quotients) + {# group.iso-class
(G Mod 𝔥 ! (m − 1)) #}
      using multisets𝔥butlast𝔎 quots𝔎notemtpy by simp
    also have ... = mset (map group.iso-class ((butlast comp𝔥.quotients) @ [last
comp𝔥.quotients]))
      using comp𝔥.quotients-butlast comp𝔥.last-quotient length𝔥big unfolding
m-def 𝔥Pm-def by auto
   also have ... = mset (map group.iso-class comp𝔥.quotients) using quots𝔥notempty
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 𝔊 = length 𝔥
proof −
 have length 𝔊 = length comp𝔊.quotients + 1 by (metis comp𝔊.quotients-length)
  also have ... = length (map group.iso-class comp𝔊.quotients) + 1 by (metis
length-map)
 also have ... = size (mset (map group.iso-class comp𝔊.quotients)) + 1 by (metis
size-mset)
 also have ... = size (mset (map group.iso-class comp𝔥.quotients)) + 1
  using jordan-hoelder-multisets is-group finite is-composition-series comp𝔥.is-composition-series
by metis
  also have ... = length (map group.iso-class comp𝔥.quotients) + 1 by (metis
size-mset)
 also have ... = length comp𝔥.quotients + 1 by (metis length-map)
 also have ... = length 𝔥 by (metis comp𝔥.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.