Secondary Sylow Theorems

Jakob von Raumer

December 17, 2016

Abstract

These theories extend the existent proof of the first sylow theorem (written by Florian Kammueller and L. C. Paulson) by what is often called the second, third and fourth sylow theorem. These theorems state propositions about the number of Sylow p-subgroups of a group and the fact that they are conjugate to each other. The proofs make use of an implementation of group actions and their properties.

Contents

theory *GroupAction*
imports
  ~~/src/HOL/Algebra/Bij

```
  ~~/src/HOL/Algebra/Sylow
begin
```

1 Group Actions

This is an implemention of group actions based on the group implementation of HOL-Algebra. An action a group G on a set M is represented by a group homomorphism between G and the group of bijections on M

1.1 Preliminaries and Definition

First, we need two theorems about singletons and sets of singletons which unfortunately are not included in the library.

```
theorem singleton-intersection:
  assumes A:card A = 1
  assumes B:card B = 1
  assumes noteq:A ≠ B
  shows A ∩ B = {}
using assms by(auto simp:card-Suc-eq)

theorem card-singleton-set:
  assumes finA:finite A
  assumes cardOne:∀ x ∈ A.(card x = 1)
  shows card (⋃ A) = card A
proof −
  from finA have card (⋃ A) = (∑ x∈A. card x)
  proof(rule card-Union-disjoint)
    from cardOne show ∀ A∈A. finite A by (auto intro: card-ge-0-finite)
  next
    show ∀ x∈A. ∀ y∈A. x ≠ y ⟶ x ∩ y = {}
    proof(clarify)
      fix x y
      assume x:x ∈ A and y:y ∈ A and x ≠ y
      with cardOne have card x = 1 card y = 1 by auto
      with ‹x ≠ y› show x ∩ y = {} by (metis singleton-intersection)
    qed
  qed
  also from cardOne have ... = card A by simp
  finally show ?thesis.
qed
```

Intersecting Cosets are equal:

```
lemma (in subgroup) repr-independence2:
  assumes group:group G
  assumes U:U ∈ rcosets_G H
  assumes g:g ∈ U
  shows U = H #> g
```

proof −
from *U* **obtain** *h* **where** *h*:$h \in$ *carrier G U* $= H$ #> *h* **unfolding** *RCOSETS-def* **by** *auto*
with *g* **have** $g \in H$ #> *h* **by** *simp*
with *group h* **show** $U = H$ #> *g* **by** (*metis group.repr-independence is-subgroup*)
qed

locale *group-action* = *group* +
fixes φ *M*
assumes *grouphom*:*group-hom G* (*BijGroup M*) φ

context *group-action*
begin

lemma *is-group-action*:*group-action G* φ *M*..

The action of **1** has no effect:

lemma *one-is-id*:
assumes $m \in M$
shows (φ **1**) $m = m$
proof −
from *grouphom* **have** (φ **1**) $m = \mathbf{1}_{(BijGroup\ M)}\ m$ **by** (*metis group-hom.hom-one*)
also have ... = ($\lambda x \in M.\ x$) *m* **unfolding** *BijGroup-def* **by** (*metis monoid.select-convs*(*2*))
also from *assms* **have** ... = *m* **by** *simp*
finally show *?thesis*.
qed

lemma *action-closed*:
assumes *m*:$m \in M$
assumes *g*:$g \in$ *carrier G*
shows $\varphi\ g\ m \in M$
using *assms grouphom group-hom.hom-closed* **unfolding** *BijGroup-def Bij-def bij-betw-def* **by** *fastforce*

lemma *img-in-bij*:
assumes $g \in$ *carrier G*
shows (φ *g*) $\in$ *Bij M*
using *assms grouphom* **unfolding** *BijGroup-def* **by** (*auto dest*: *group-hom.hom-closed*)

The action of *inv g* reverts the action of *g*

lemma *group-inv-rel*:
assumes *g*:$g \in$ *carrier G*
assumes *mn*:$m \in M$ $n \in M$
assumes *phi*:(φ *g*) $n = m$
shows (φ (*inv g*)) $m = n$
proof −
from *g* **have** *bij*:(φ *g*) $\in$ *Bij M* **unfolding** *BijGroup-def* **by** (*metis img-in-bij*)
with *g grouphom* **have** φ (*inv g*) = *restrict* (*inv-into M* (φ *g*)) *M* **by**(*metis inv-BijGroup group-hom.hom-inv*)

hence φ (*inv g*) m = (*restrict* (*inv-into M* (φ *g*)) *M*) *m* **by** *simp*
also from *mn* **have** ... = (*inv-into M* (φ *g*)) *m* **by** (*metis restrict-def*)
also from *g phi* **have** ... = (*inv-into M* (φ *g*)) ((φ *g*) *n*) **by** *simp*
also from ‹φ *g* $\in$ *Bij M*› *Bij-def* **have** *bij-betw* (φ *g*) *M M* **by** *auto*
hence *inj-on* (φ *g*) *M* **by** (*metis bij-betw-imp-inj-on*)
with *g mn* **have** (*inv-into M* (φ *g*)) ((φ *g*) *n*) = *n* **by** (*metis inv-into-f-f*)
finally show φ (*inv g*) *m* = *n*.
qed

lemma *images-are-bij*:
assumes *g*:*g* $\in$ *carrier G*
shows *bij-betw* (φ *g*) *M M*
proof −
from *g* **have** *bij*:(φ *g*) $\in$ *Bij M* **unfolding** *BijGroup-def* **by** (*metis img-in-bij*)
with *Bij-def* **show** *bij-betw* (φ *g*) *M M* **by** *auto*
qed

lemma *action-mult*:
assumes *g*:*g* $\in$ *carrier G*
assumes *h*:*h* $\in$ *carrier G*
assumes *m*:*m* $\in$ *M*
shows (φ *g*) ((φ *h*) *m*) = (φ (*g* $\otimes$ *h*)) *m*
proof −
from *g* **have** φg:(φ *g*) $\in$ *Bij M* **unfolding** *BijGroup-def* **by** (*rule img-in-bij*)
from *h* **have** φh:(φ *h*) $\in$ *Bij M* **unfolding** *BijGroup-def* **by** (*rule img-in-bij*)
from *h* **have** *bij-betw* (φ *h*) *M M* **by** (*rule images-are-bij*)
hence (φ *h*) ‘ *M* = *M* **by** (*metis bij-betw-def*)
with *m* **have** *hm*:(φ *h*) *m* $\in$ *M* **by** (*metis imageI*)
from *grouphom g h* **have** (φ (*g* $\otimes$ *h*)) = ((φ *g*) $\otimes_{(BijGroup\ M)}$ (φ *h*)) **by** (*rule group-hom.hom-mult*)
hence φ (*g* $\otimes$ *h*) *m* = ((φ *g*) $\otimes_{(BijGroup\ M)}$ (φ *h*)) *m* **by** *simp*
also from φg φh **have** ... = (*compose M* (φ *g*) (φ *h*)) *m* **unfolding** *BijGroup-def* **by** *simp*
also from φg φh *hm* **have** ... = (φ *g*) ((φ *h*) *m*) **by** (*metis compose-eq m*)
finally show (φ *g*) ((φ *h*) *m*) = (φ (*g* $\otimes$ *h*)) *m*..
qed

1.2 The orbit relation

The following describes the relation containing the information whether two elements of M lie in the same orbit of the action

definition *same-orbit-rel*
where *same-orbit-rel* = $\{p \in M \times M.\ \exists g \in$ *carrier G*. (φ *g*) (*snd p*) = (*fst p*)$\}$

Use the library about equivalence relations to define the set of orbits and the map assigning to each element of M its orbit

definition *orbits*
where *orbits* = *M* // *same-orbit-rel*

definition *orbit* :: $'c \Rightarrow 'c$ *set*
  where *orbit* m = *same-orbit-rel* ‘‘ $\{m\}$

Next, we define a more easy-to-use characterization of an orbit.

lemma *orbit-char*:
  assumes $m{:}m \in M$
  shows *orbit* $m = \{n.\ \exists g.\ g \in$ *carrier* $G \wedge (\varphi\ g)\ m = n\}$
using *assms* **unfolding** *orbit-def Image-def same-orbit-rel-def*
proof(*auto*)
  fix $x\ g$
  assume $g{:}g \in$ *carrier* G **and** $\varphi\ g\ x \in M\ x \in M$
  hence φ (*inv* g) $(\varphi\ g\ x) = x$ **by** (*metis group-inv-rel*)
  moreover from g **have** *inv* $g \in$ *carrier* G **by** (*rule inv-closed*)
  ultimately show $\exists h.\ h \in$ *carrier* $G \wedge \varphi\ h\ (\varphi\ g\ x) = x$ **by** *auto*
next
  fix g
  assume $g{:}g \in$ *carrier* G
  with m **show** $\varphi\ g\ m \in M$ **by** (*metis action-closed*)
  with $m\ g$ **have** φ (*inv* g) $(\varphi\ g\ m) = m$ **by** (*metis group-inv-rel*)
  moreover from g **have** *inv* $g \in$ *carrier* G **by** (*rule inv-closed*)
  ultimately show $\exists h{\in}$*carrier* $G.\ \varphi\ h\ (\varphi\ g\ m) = m$ **by** *auto*
qed

lemma *same-orbit-char*:
  assumes $m \in M\ n \in M$
  shows $(m,\ n) \in$ *same-orbit-rel* $= (\exists g \in$ *carrier* $G.\ ((\varphi\ g)\ n = m))$
unfolding *same-orbit-rel-def* **using** *assms* **by** *auto*

Now we show that the relation we've defined is, indeed, an equivalence relation:

lemma *same-orbit-is-equiv*:
  shows *equiv M same-orbit-rel*
proof(*rule equivI*)
  show *refl-on M same-orbit-rel*
  proof(*rule refl-onI*)
    show *same-orbit-rel* $\subseteq M \times M$ **unfolding** *same-orbit-rel-def* **by** *auto*
  next
    fix m
    assume $m \in M$
    hence $(\varphi\ \mathbf{1})\ m = m$ **by**(*rule one-is-id*)
    with ‹$m \in M$› **show** $(m,\ m) \in$ *same-orbit-rel* **unfolding** *same-orbit-rel-def*
by (*auto simp*:*same-orbit-char*)
  qed
next
  show *sym same-orbit-rel*
  proof(*rule symI*)
    fix $m\ n$
    assume $mn{:}(m,\ n) \in$ *same-orbit-rel*

then obtain g where g:$g \in$ *carrier* G φ g n = m **unfolding** *same-orbit-rel-def* **by** *auto*
hence *invg*:*inv* $g \in$ *carrier* G **by** (*metis inv-closed*)
from *mn* **have** $(m, n) \in M \times M$ **unfolding** *same-orbit-rel-def* **by** *simp*
hence *mn2*:$m \in M$ $n \in M$ **by** *auto*
from g *mn2* **have** φ (*inv* g) $m = n$ **by** (*metis group-inv-rel*)
with *invg mn2* **show** $(n, m) \in$ *same-orbit-rel* **unfolding** *same-orbit-rel-def* **by** *auto*
qed
next
show *trans same-orbit-rel*
proof(*rule transI*)
fix x y z
assume *xy*:$(x, y) \in$ *same-orbit-rel*
then obtain g **where** g:$g \in$ *carrier* G **and** *grel*:$(\varphi\ g)\ y = x$ **unfolding** *same-orbit-rel-def* **by** *auto*
assume *yz*:$(y, z) \in$ *same-orbit-rel*
then obtain h **where** h:$h \in$ *carrier* G **and** *hrel*:$(\varphi\ h)\ z = y$ **unfolding** *same-orbit-rel-def* **by** *auto*
from g h **have** *gh*:$g \otimes h \in$ *carrier* G **by** *simp*
from *xy yz* **have** $x \in M$ $z \in M$ **unfolding** *same-orbit-rel-def* **by** *auto*
with g h **have** $\varphi\ (g \otimes h)\ z = (\varphi\ g)\ ((\varphi\ h)\ z)$ **by** (*metis action-mult*)
also from *hrel grel* **have** ... $= x$ **by** *simp*
finally have $\varphi\ (g \otimes h)\ z = x$**.**
with *gh* ‹$x \in M$› ‹$z \in M$› **show** $(x, z) \in$ *same-orbit-rel* **unfolding** *same-orbit-rel-def* **by** *auto*
qed
qed

1.3 Stabilizer and fixed points

The following definition models the stabilizer of a group action:

definition *stabilizer* :: $'c \Rightarrow$ -
where *stabilizer* $m = \{g \in$ *carrier* $G.\ (\varphi\ g)\ m = m\}$

This shows that the stabilizer of m is a subgroup of G.

lemma *stabilizer-is-subgroup*:
assumes m:$m \in M$
shows *subgroup* (*stabilizer* m) G
proof(*rule subgroupI*)
show *stabilizer* $m \subseteq$ *carrier* G **unfolding** *stabilizer-def* **by** *auto*
next
from m **have** $(\varphi\ \mathbf{1})\ m = m$ **by** (*rule one-is-id*)
hence $\mathbf{1} \in$ *stabilizer* m **unfolding** *stabilizer-def* **by** *simp*
thus *stabilizer* $m \neq \{\}$ **by** *auto*
next
fix g
assume g:$g \in$ *stabilizer* m
hence $g \in$ *carrier* G $(\varphi\ g)\ m = m$ **unfolding** *stabilizer-def* **by** *simp+*

with *m* **have** *ginv*:(φ (*inv g*)) *m* = *m* **by** (*metis group-inv-rel*)
from ‹*g* ∈ *carrier G*› **have** *inv g* ∈ *carrier G* **by** (*metis inv-closed*)
with *ginv* **show** (*inv g*) ∈ *stabilizer m* **unfolding** *stabilizer-def* **by** *simp*
next
fix *g h*
assume *g*:*g* ∈ *stabilizer m*
hence *g2*:*g* ∈ *carrier G* **unfolding** *stabilizer-def* **by** *simp*
assume *h*:*h* ∈ *stabilizer m*
hence *h2*:*h* ∈ *carrier G* **unfolding** *stabilizer-def* **by** *simp*
with *g2* **have** *gh*:*g* ⊗ *h* ∈ *carrier G* **by** (*rule m-closed*)
from *g2 h2 m* **have** φ (*g* ⊗ *h*) *m* = (φ *g*) ((φ *h*) *m*) **by** (*metis action-mult*)
also from *g h* **have** ... = *m* **unfolding** *stabilizer-def* **by** *simp*
finally have φ (*g* ⊗ *h*) *m* = *m*.
with *gh* **show** *g* ⊗ *h* ∈ *stabilizer m* **unfolding** *stabilizer-def* **by** *simp*
qed

Next, we define and characterize the fixed points of a group action.

definition *fixed-points* :: *'c set*
where *fixed-points* = {*m* ∈ *M*. *carrier G* ⊆ *stabilizer m*}

lemma *fixed-point-char*:
assumes *m* ∈ *M*
shows (*m* ∈ *fixed-points*) = (∀ *g*∈*carrier G*. φ *g m* = *m*)
using *assms* **unfolding** *fixed-points-def stabilizer-def* **by** *force*

lemma *orbit-contains-rep*:
assumes *m*:*m* ∈ *M*
shows *m* ∈ *orbit m*
unfolding *orbit-def* **using** *assms* **by** (*metis equiv-class-self same-orbit-is-equiv*)

lemma *singleton-orbit-eq-fixed-point*:
assumes *m*:*m* ∈ *M*
shows (*card* (*orbit m*) = *1*) = (*m* ∈ *fixed-points*)
proof
assume *card*:*card* (*orbit m*) = *1*
from *m* **have** *m* ∈ *orbit m* **by** (*rule orbit-contains-rep*)
from *m* **show** *m* ∈ *fixed-points* **unfolding** *fixed-points-def*
proof(*auto*)
fix *g*
assume *gG*:*g* ∈ *carrier G*
with *m* **have** φ *g m* ∈ *orbit m* **by** (*auto dest*:*orbit-char*)
with ‹*m* ∈ *orbit m*› *card* **have** φ *g m* = *m* **by** (*auto simp add*: *card-Suc-eq*)
with *gG* **show** *g* ∈ *stabilizer m* **unfolding** *stabilizer-def* **by** *simp*
qed
next
assume *m* ∈ *fixed-points*
hence *fixed*:*carrier G* ⊆ *stabilizer m* **unfolding** *fixed-points-def* **by** *simp*
from *m* **have** *orbit m* = {*m*}
proof(*auto simp add*: *orbit-contains-rep*)

fix n
assume $n \in orbit\ m$
with m **obtain** g **where** $g{:}g \in carrier\ G\ \varphi\ g\ m = n$ **by** $(auto\ dest{:}\ orbit\text{-}char)$
moreover with *fixed* **have** $\varphi\ g\ m = m$ **unfolding** *stabilizer-def* **by** *auto*
ultimately show $n = m$ **by** *simp*
qed
thus $card\ (orbit\ m) = 1$ **by** *simp*
qed

1.4 The Orbit-Stabilizer Theorem

This section contains some theorems about orbits and their quotient groups. The first one is the well-known orbit-stabilizer theorem which establishes a bijection between the the quotient group of the an element's stabilizer and its orbit.

theorem *orbit-thm*:
assumes $m{:}m \in M$
assumes $rep{:}\bigwedge U.\ U \in (carrier\ (G\ Mod\ (stabilizer\ m))) \Longrightarrow rep\ U \in U$
shows $bij\text{-}betw\ (\lambda H.\ (\varphi\ (inv\ (rep\ H))\ m))\ (carrier\ (G\ Mod\ (stabilizer\ m)))\ (orbit\ m)$
apply$(simp\ add{:}bij\text{-}betw\text{-}def)$
proof$(auto)$
show $inj\text{-}on\ (\lambda H.\ \varphi\ (inv\ (rep\ H))\ m)\ (carrier\ (G\ Mod\ stabilizer\ m))$
proof$(rule\ inj\text{-}onI)$
fix $U\ V$
assume $U{:}U \in carrier\ (G\ Mod\ (stabilizer\ m))$
assume $V{:}V \in carrier\ (G\ Mod\ (stabilizer\ m))$
def $h \equiv rep\ V$
def $g \equiv rep\ U$
have $stabSubset{:}(stabilizer\ m) \subseteq carrier\ G$ **unfolding** *stabilizer-def* **by** *auto*
from m **have** $stabSubgroup{:}\ subgroup\ (stabilizer\ m)\ G$ **by** $(metis\ stabilizer\text{-}is\text{-}subgroup)$
from $V\ rep$ **have** $hV{:}h \in V$ **unfolding** *h-def* **by** *simp*
from $V\ stabSubset$ **have** $V \subseteq carrier\ G$ **unfolding** *FactGroup-def RCOSETS-def r-coset-def* **by** *auto*
with hV **have** $hG{:}h \in carrier\ G$ **by** *auto*
hence $hinvG{:}inv\ h \in carrier\ G$ **by** $(metis\ inv\text{-}closed)$
from $U\ rep$ **have** $gU{:}g \in U$ **unfolding** *g-def* **by** *simp*
from $U\ stabSubset$ **have** $U \subseteq carrier\ G$ **unfolding** *FactGroup-def RCOSETS-def r-coset-def* **by** *auto*
with gU **have** $gG{:}g \in carrier\ G$ **by** *auto*
hence $ginvG{:}inv\ g \in carrier\ G$ **by** $(metis\ inv\text{-}closed)$
from $gG\ hinvG$ **have** $ginvhG{:}\ g \otimes inv\ h \in carrier\ G$ **by** $(metis\ m\text{-}closed)$
assume $reps{:}\varphi\ (inv\ rep\ U)\ m = \varphi\ (inv\ rep\ V)\ m$
hence $gh{:}\varphi\ (inv\ g)\ m = \varphi\ (inv\ h)\ m$ **unfolding** *g-def h-def* **.**
from $gG\ hinvG\ m$ **have** $\varphi\ (g \otimes (inv\ h))\ m = \varphi\ g\ (\varphi\ (inv\ h)\ m)$ **by** $(metis\ action\text{-}mult)$
also with $gh\ ginvG\ gG\ m$ **have** $... = \varphi\ (g \otimes inv\ g)\ m$ **by** $(metis\ action\text{-}mult)$
also with $m\ gG$ **have** $... = m$ **by** $(auto\ simp{:}\ one\text{-}is\text{-}id)$

finally have φ (*g* ⊗ *inv h*) *m* = *m*.
with *ginvhG* **have** (*g* ⊗ *inv h*) ∈ *stabilizer m* **unfolding** *stabilizer-def* **by** *simp*
hence (*stabilizer m*) #> (*g* ⊗ *inv h*) = (*stabilizer m*) #> **1** **by** (*metis coset-join2 coset-mult-one m stabSubset stabilizer-is-subgroup subgroup.mem-carrier*)
with *hinvG hG gG stabSubset* **have** *stabgstabh*:(*stabilizer m*) #> *g* = (*stabilizer m*) #> *h* **by** (*metis coset-mult-inv1 group.coset-mult-one is-group*)
from *stabSubgroup is-group U gU* **have** *U* = (*stabilizer m*) #> *g* **unfolding** *FactGroup-def* **by** (*simp add*:*subgroup.repr-independence2*)
also with *stabgstabh is-group stabSubgroup V hV subgroup.repr-independence2* **have** ... = *V* **unfolding** *FactGroup-def* **by** *force*
finally show *U* = *V*.
qed
next
have *stabSubset*:*stabilizer m* ⊆ *carrier G* **unfolding** *stabilizer-def* **by** *auto*
fix *H*
assume *H*:*H* ∈ *carrier* (*G Mod stabilizer m*)
with *rep* **have** *rep H* ∈ *H* **by** *simp*
moreover with *H stabSubset* **have** *H* ⊆ *carrier G* **unfolding** *FactGroup-def RCOSETS-def r-coset-def* **by** *auto*
ultimately have *rep H* ∈ *carrier G*..
hence *inv rep H* ∈ *carrier G* **by** (*rule inv-closed*)
with *m* **show** φ (*inv rep H*) *m* ∈ *orbit m* **by** (*auto dest*:*orbit-char*)
next
fix *n*
assume *n* ∈ *orbit m*
with *m* **obtain** *g* **where** *g*:*g* ∈ *carrier G* φ *g m* = *n* **by** (*auto dest*:*orbit-char*)
hence *invg*:*inv g* ∈ *carrier G* **by** *simp*
hence *stabinvg*:((*stabilizer m*) #> (*inv g*)) ∈ *carrier* (*G Mod stabilizer m*) **unfolding** *FactGroup-def RCOSETS-def* **by** *auto*
hence *rep* ((*stabilizer m*) #> (*inv g*)) ∈ (*stabilizer m*) #> (*inv g*) **by** (*metis rep*)
then obtain *h* **where** *h*:*h* ∈ *stabilizer m rep* ((*stabilizer m*) #> (*inv g*)) = *h* ⊗ (*inv g*) **unfolding** *r-coset-def* **by** *auto*
with *g* **have** φ (*inv rep* ((*stabilizer m*) #> (*inv g*))) *m* = φ (*inv* (*h* ⊗ (*inv g*))) *m* **by** *simp*
also from *h* **have** *hG*:*h* ∈ *carrier G* **unfolding** *stabilizer-def* **by** *simp*
with *g* **have** φ (*inv* (*h* ⊗ (*inv g*))) *m* = φ (*g* ⊗ (*inv h*)) *m* **by** (*metis inv-closed inv-inv inv-mult-group*)
also from *g hG m* **have** ... = φ *g* (φ (*inv h*) *m*) **by** (*metis action-mult inv-closed*)
also from *h m* **have** *inv h* ∈ *stabilizer m* **by** (*metis stabilizer-is-subgroup subgroup.m-inv-closed*)
hence φ *g* (φ (*inv h*) *m*) = φ *g m* **unfolding** *stabilizer-def* **by** *simp*
also from *g* **have** ... = *n* **by** *simp*
finally have *n* = φ (*inv rep* ((*stabilizer m*) #> (*inv g*))) *m*..
with *stabinvg* **show** *n* ∈ (λ*H*. φ (*inv rep H*) *m*) ‘ *carrier* (*G Mod stabilizer m*) **by** *simp*
qed

In the case of G being finite, the last theorem can be reduced to a statement about the cardinality of orbit and stabilizer:

corollary *orbit-size*:
  assumes *fin*:*finite* (*carrier G*)
  assumes *m*:$m \in M$
  shows *order G* = *card* (*orbit m*) $*$ *card* (*stabilizer m*)
proof −
  def *rep* $\equiv \lambda U \in$ (*carrier* (*G Mod* (*stabilizer m*))). (*SOME x*. $x \in U$)
  have $\bigwedge U$. $U \in$ (*carrier* (*G Mod* (*stabilizer m*))) $\Longrightarrow$ *rep U* $\in U$
  proof −
    fix *U*
    assume *U*:$U \in$ *carrier* (*G Mod stabilizer m*)
    then obtain *g* **where** $g \in$ *carrier G* *U* = (*stabilizer m*) #> *g* **unfolding** *FactGroup-def RCOSETS-def* **by** *auto*
    with *m* **have** (*SOME x*. $x \in U$) $\in U$ **by** (*metis rcos-self stabilizer-is-subgroup someI-ex*)
    with *U* **show** *rep U* $\in U$ **unfolding** *rep-def* **by** *simp*
  qed
  with *m* **have** *bij*:*card* (*carrier* (*G Mod* (*stabilizer m*))) = *card* (*orbit m*) **by** (*metis bij-betw-same-card orbit-thm*)
  from *fin m* **have** *card* (*carrier* (*G Mod* (*stabilizer m*))) $*$ *card* (*stabilizer m*) = *order G* **unfolding** *FactGroup-def* **by** (*simp add*: *stabilizer-is-subgroup lagrange*)
  with *bij* **show** *?thesis* **by** *simp*
qed

lemma *orbit-not-empty*:
  assumes *fin*:*finite M*
  assumes *A*:$A \in$ *orbits*
  shows *card A* > *0*
proof −
  from *A* **obtain** *m* **where** $m \in M$ *A* = *orbit m* **unfolding** *orbits-def quotient-def orbit-def* **by** *auto*
  hence $m \in A$ **by** (*metis orbit-contains-rep*)
  hence $A \neq \{\}$ **unfolding** *orbits-def* **by** *auto*
  moreover from *fin A* **have** *finite A* **unfolding** *orbits-def quotient-def Image-def same-orbit-rel-def* **by** *auto*
  ultimately show *?thesis* **by** *auto*
qed

lemma *fin-set-imp-fin-orbits*:
  assumes *finM*:*finite M*
  shows *finite orbits*
using *assms* **unfolding** *orbits-def quotient-def* **by** *simp*

lemma *singleton-orbits*:
  shows $\bigcup\{N \in$ *orbits*. *card N* = *1*$\}$ = *fixed-points*
proof
  show $\bigcup\{N \in$ *orbits*. *card N* = *1*$\} \subseteq$ *fixed-points*
  proof

fix x
assume $a{:}x \in \bigcup\{N \in orbits.\ card\ N = 1\}$
hence $x \in M$ **unfolding** *orbits-def quotient-def Image-def same-orbit-rel-def* **by** *auto*
from a **obtain** N **where** $N{:}N \in orbits\ card\ N = 1\ x \in N$ **by** *auto*
then obtain y **where** $Norbit{:}N = orbit\ y\ y \in M$ **unfolding** *orbits-def quotient-def orbit-def* **by** *auto*
hence $y \in N$ **by** (*metis orbit-contains-rep*)
with N **have** $Nsing{:}N = \{x\}\ N = \{y\}$ **by** (*auto simp*: *card-Suc-eq*)
hence $x = y$ **by** *simp*
with *Norbit* **have** $Norbit2{:}N = orbit\ x$ **by** *simp*
have $\{g \in carrier\ G.\ \varphi\ g\ x = x\} = carrier\ G$
proof(*auto*)
fix g
assume $g \in carrier\ G$
with ‹$x \in M$› **have** $\varphi\ g\ x \in orbit\ x$ **by** (*auto dest*:*orbit-char*)
with *Nsing* **show** $\varphi\ g\ x = x$ **by** (*metis Norbit2 singleton-iff*)
qed
with ‹$x \in M$› **show** $x \in$ *fixed-points* **unfolding** *fixed-points-def stabilizer-def* **by** *simp*
qed
next
show *fixed-points* $\subseteq \bigcup\{N \in orbits.\ card\ N = 1\}$
proof
fix m
assume $m{:}m \in$ *fixed-points*
hence $mM{:}m \in M$ **unfolding** *fixed-points-def* **by** *simp*
hence $orbit{:}orbit\ m \in orbits$ **unfolding** *orbits-def quotient-def orbit-def* **by** *auto*
from $mM\ m$ **have** $card\ (orbit\ m) = 1$ **by** (*metis singleton-orbit-eq-fixed-point*)
with *orbit* **have** $orbit\ m \in \{N \in orbits.\ card\ N = 1\}$ **by** *simp*
with mM **show** $m \in \bigcup\{N \in orbits.\ card\ N = 1\}$ **by** (*auto dest*: *orbit-contains-rep*)
qed
qed

If G is a p-group acting on a finite set, a given orbit is either a singleton or p divides its cardinality.

lemma *p-dvd-orbit-size*:
assumes $orderG{:}order\ G = p\ \hat{}\ a$
assumes $prime{:}prime\ p$
assumes $finM{:}finite\ M$
assumes $Norbit{:}N \in orbits$
assumes $card\ N > 1$
shows $p\ dvd\ card\ N$
proof −
from *Norbit* **obtain** m **where** $m{:}m \in M\ N = orbit\ m$ **unfolding** *orbits-def quotient-def orbit-def* **by** *auto*
from *prime* **have** $0 < p\ \hat{}\ a$ **by** (*simp add*: *prime-gt-0-nat*)
with *orderG* **have** *finite* (*carrier G*) **unfolding** *order-def* **by** (*metis card-infinite*

less-nat-zero-code)
with *m* **have** *order G* = *card* (*orbit m*) ∗ *card* (*stabilizer m*) **by** (*metis orbit-size*)
with *orderG m* **have** *p* ˆ *a* = *card N* ∗ *card* (*stabilizer m*) **by** *simp*
with ‹*card N* > *1*› **show** *?thesis*
by (*metis dvd-mult2 dvd-mult-cancel1 nat-dvd-not-less nat-mult-1 prime prime-dvd-power-nat prime-factor-nat prime-nat-iff zero-less-one*)
qed

As a result of the last lemma the only orbits that count modulo p are the fixed points

lemma *fixed-point-congruence*:
assumes *order G* = *p* ˆ *a*
assumes *prime p*
assumes *finM*:*finite M*
shows *card M mod p* = *card fixed-points mod p*
proof −
def *big-orbits* ≡ {*N*∈*orbits*. *card N* > *1*}
from *finM* **have** *orbit-part*:*orbits* = *big-orbits* ∪ {*N*∈*orbits*. *card N* = *1*} **unfolding** *big-orbits-def* **by** (*auto dest*:*orbit-not-empty*)
have *orbit-disj*:*big-orbits* ∩ {*N*∈*orbits*. *card N* = *1*} = {} **unfolding** *big-orbits-def* **by** *auto*
from *finM* **have** *orbits-fin*:*finite orbits* **by** (*rule fin-set-imp-fin-orbits*)
hence *fin-parts*:*finite big-orbits finite* {*N*∈*orbits*. *card N* = *1*} **unfolding** *big-orbits-def* **by** *simp*+
from *assms* **have** ⋀*N*. *N* ∈ *big-orbits* ⟹ *p dvd card N* **unfolding** *big-orbits-def* **by** (*auto simp*: *p-dvd-orbit-size*)
hence *orbit-div*:⋀*N*. *N* ∈ *big-orbits* ⟹ *card N* = (*card N div p*) ∗ *p* **by** (*metis dvd-mult-div-cancel mult.commute*)
have *card M* = *card* (⋃ *orbits*) **unfolding** *orbits-def* **by** (*metis Union-quotient same-orbit-is-equiv*)
also from *orbits-fin* **have** *card* (⋃ *orbits*) = (∑ *N*∈*orbits*. *card N*) **unfolding** *orbits-def*
apply(*rule card-Union-disjoint*)
defer *1*
apply(*metis same-orbit-is-equiv quotient-disj*)
using *finM same-orbit-rel-def* **apply**(*auto dest*:*finite-equiv-class*)
done
also from *orbit-part orbit-disj fin-parts* **have** ... = (∑ *N*∈*big-orbits*. *card N*) + (∑ *N*∈{*N'*∈*orbits*. *card N'* = *1*}. *card N*) **by** (*metis* (*lifting*) *sum.union-disjoint*)
also from *assms orbit-div fin-parts* **have** ... = (∑ *N*∈*big-orbits*. (*card N div p*) ∗ *p*) + *card* (⋃{*N'*∈*orbits*. *card N'* = *1*}) **by** (*auto simp*: *card-singleton-set*)
also have ... = (∑ *N*∈*big-orbits*. *card N div p*) ∗ *p* + *card fixed-points* **using** *singleton-orbits* **by** (*auto simp*:*sum-distrib-right*)
finally have *card M* = (∑ *N*∈*big-orbits*. *card N div p*) ∗ *p* + *card fixed-points***.**
hence *card M mod p* = ((∑ *N*∈*big-orbits*. *card N div p*) ∗ *p* + *card fixed-points*) *mod p* **by** *simp*
also have ... = (*card fixed-points*) *mod p* **by** (*metis mod-mult-self3*)
finally show *?thesis***.**
qed

We can restrict any group action to the action of a subgroup:

lemma *subgroup-action*:
  assumes H:*subgroup* H G
  shows *group-action* $(G(\!|$*carrier* $:= H|\!))$ φ M
unfolding *group-action-def group-action-axioms-def group-hom-def group-hom-axioms-def hom-def*
using *assms*
proof (*auto simp add*: *is-group subgroup.subgroup-is-group group-BijGroup*)
  fix x
  assume $x \in H$
  with H **have** $x \in$ *carrier* G **by** (*metis subgroup.mem-carrier*)
  with *grouphom* **show** φ $x \in$ *carrier* (*BijGroup* M) **by** (*metis group-hom.hom-closed*)
next
  fix x y
  assume x:$x \in H$ **and** y:$y \in H$
  with H **have** $x \in$ *carrier* G $y \in$ *carrier* G **by** (*metis subgroup.mem-carrier*)+
  with *grouphom* **show** φ $(x \otimes y) = \varphi$ $x \otimes_{BijGroup\ M} \varphi$ y **by** (*simp add*: *group-hom.hom-mult*)
qed

end

1.5 Some Examples for Group Actions

lemma (**in** *group*) *right-mult-is-bij*:
  assumes h:$h \in$ *carrier* G
  shows $(\lambda g \in$ *carrier* $G.$ $h \otimes g) \in$ *Bij* (*carrier* G)
proof(*auto simp add*:*Bij-def bij-betw-def inj-on-def*)
  fix x y
  assume x:$x \in$ *carrier* G **and** y:$y \in$ *carrier* G **and** $h \otimes x = h \otimes y$
  with h **show** $x = y$ **by** (*metis l-cancel*)
next
  fix x
  assume x:$x \in$ *carrier* G
  with h **show** $h \otimes x \in$ *carrier* G **by** (*metis m-closed*)
  from x h **have** *inv* $h \otimes x \in$ *carrier* G **by** (*metis m-closed inv-closed*)
  moreover from x h **have** $h \otimes$ (*inv* $h \otimes x) = x$ **by** (*metis inv-closed r-inv m-assoc l-one*)
  ultimately show $x \in$ *op* $\otimes$ h ‘ *carrier* G **by** *force*
qed

lemma (**in** *group*) *right-mult-group-action*:
  shows *group-action* G $(\lambda h.$ $\lambda g \in$ *carrier* $G.$ $h \otimes g)$ (*carrier* G)
unfolding *group-action-def group-action-axioms-def group-hom-def group-hom-axioms-def hom-def*
proof(*auto simp add*:*is-group group-BijGroup*)
  fix h
  assume $h \in$ *carrier* G
  thus $(\lambda g \in$ *carrier* $G.$ $h \otimes g) \in$ *carrier* (*BijGroup* (*carrier* G)) **unfolding**

BijGroup-def **by** (*auto simp*:*right-mult-is-bij*)
next
  fix *x y*
  assume *x*:*x* ∈ *carrier G* **and** *y*:*y* ∈ *carrier G*
  def *multx* ≡ (λ*g*∈*carrier G*. *x* ⊗ *g*) **and** *multy* ≡ (λ*g*∈*carrier G*. *y* ⊗ *g*)
  with *x y* **have** *multx* ∈ (*Bij* (*carrier G*)) *multy* ∈ (*Bij* (*carrier G*)) **by** (*metis right-mult-is-bij*)+
  hence *multx* $\otimes_{BijGroup\ (carrier\ G)}$ *multy* = (λ*g*∈*carrier G*. *multx* (*multy g*)) **unfolding** *BijGroup-def* **by** (*auto simp*: *compose-def*)
  also have ... = (λ*g*∈*carrier G*. (*x* ⊗ *y*) ⊗ *g*) **unfolding** *multx-def multy-def*
  proof(*rule restrict-ext*)
    fix *g*
    assume *g*:*g* ∈ *carrier G*
    with *x y* **have** *x* ⊗ *y* ∈ *carrier G* *y* ⊗ *g* ∈ *carrier G* **by** *simp*+
    with *x y g* **show** (λ*g*∈*carrier G*. *x* ⊗ *g*) ((λ*g*∈*carrier G*. *y* ⊗ *g*) *g*) = *x* ⊗ *y* ⊗ *g* **by** (*auto simp*:*m-assoc*)
  qed
  finally show (λ*g*∈*carrier G*. (*x* ⊗ *y*) ⊗ *g*) = (λ*g*∈*carrier G*. *x* ⊗ *g*) $\otimes_{BijGroup\ (carrier\ G)}$ (λ*g*∈*carrier G*. *y* ⊗ *g*) **unfolding** *multx-def multy-def* **by** *simp*
qed

lemma (**in** *group*) *rcosets-closed*:
  assumes *HG*:*subgroup H G*
  assumes *g*:*g* ∈ *carrier G*
  assumes *M*:*M* ∈ *rcosets H*
  shows *M* #> *g* ∈ *rcosets H*
proof −
  from *M* **obtain** *h* **where** *h*:*h* ∈ *carrier G M* = *H* #> *h* **unfolding** *RCOSETS-def* **by** *auto*
  with *g HG* **have** *M* #> *g* = *H* #> (*h* ⊗ *g*) **by** (*metis coset-mult-assoc subgroup-imp-subset*)
  with *HG g h* **show** *M* #> *g* ∈ *rcosets H* **by** (*metis rcosetsI subgroup.m-closed subgroup-imp-subset subgroup-self*)
qed

lemma (**in** *group*) *inv-mult-on-rcosets-is-bij*:
  assumes *HG*:*subgroup H G*
  assumes *g*:*g* ∈ *carrier G*
  shows (λ*U* ∈ *rcosets H*. *U* #> *inv g*) ∈ *Bij* (*rcosets H*)
proof(*auto simp add*:*Bij-def bij-betw-def inj-on-def*)
  fix *M*
  assume *M* ∈ *rcosets H*
  with *HG g* **show** *M* #> *inv g* ∈ *rcosets H* **by** (*metis inv-closed rcosets-closed*)
next
  fix *M*
  assume *M*:*M* ∈ *rcosets H*
  with *HG g* **have** *M* #> *g* ∈ *rcosets H* **by** (*rule rcosets-closed*)
  moreover from *M HG g* **have** *M* #> *g* #> *inv g* = *M* **by** (*metis coset-mult-assoc coset-mult-inv2 inv-closed is-group subgroup.rcosets-carrier*)

ultimately show $M \in (\lambda U.\ U \ \#{>}\ inv\ g)$ ‘ $(rcosets\ H)$ **by** *auto*
next
fix *M N x*
assume $M{:}M \in rcosets\ H$ **and** $N{:}N \in rcosets\ H$ **and** $M\ \#{>}\ inv\ g = N\ \#{>}\ inv\ g$
hence $(M\ \#{>}\ inv\ g)\ \#{>}\ g = (N\ \#{>}\ inv\ g)\ \#{>}\ g$ **by** *simp*
with *HG M N g* **have** $M\ \#{>}\ (inv\ g \otimes g) = N\ \#{>}\ (inv\ g \otimes g)$ **by** (*metis coset-mult-assoc is-group subgroup.m-inv-closed subgroup.rcosets-carrier subgroup-self*)
with *HG M N g* **have** $a1{:}M = N$ **by** (*metis l-inv coset-mult-one is-group subgroup.rcosets-carrier*)
{
assume $x \in M$
with *a1* **show** $x \in N$ **by** *simp*
}
{
assume $x \in N$
with *a1* **show** $x \in M$ **by** *simp*
}
qed

lemma (**in** *group*) *inv-mult-on-rcosets-action*:
assumes *HG*:*subgroup H G*
shows *group-action* $G\ (\lambda g.\ \lambda U \in rcosets\ H.\ U\ \#{>}\ inv\ g)\ (rcosets\ H)$
unfolding *group-action-def group-action-axioms-def group-hom-def group-hom-axioms-def hom-def*
proof(*auto simp add*:*is-group group-BijGroup*)
fix *h*
assume $h \in carrier\ G$
with *HG* **show** $(\lambda U \in rcosets\ H.\ U\ \#{>}\ inv\ h) \in carrier\ (BijGroup\ (rcosets\ H))$ **unfolding** *BijGroup-def* **by** (*auto simp*:*inv-mult-on-rcosets-is-bij*)
next
fix *x y*
assume $x{:}x \in carrier\ G$ **and** $y{:}y \in carrier\ G$
def $cosx \equiv (\lambda U{\in}rcosets\ H.\ U\ \#{>}\ inv\ x)$ **and** $cosy \equiv (\lambda U{\in}rcosets\ H.\ U\ \#{>}\ inv\ y)$
with *x y HG* **have** $cosx \in (Bij\ (rcosets\ H))\ cosy \in (Bij\ (rcosets\ H))$ **by** (*metis inv-mult-on-rcosets-is-bij*)+
hence $cosx \otimes_{BijGroup\ (rcosets\ H)} cosy = (\lambda U{\in}rcosets\ H.\ cosx\ (cosy\ U))$ **unfolding** *BijGroup-def* **by** (*auto simp*: *compose-def*)
also have ... $= (\lambda U{\in}rcosets\ H.\ U\ \#{>}\ inv\ (x \otimes y))$ **unfolding** *cosx-def cosy-def*
proof(*rule restrict-ext*)
fix *U*
assume $U{:}U \in rcosets\ H$
with *HG y* **have** $U\ \#{>}\ inv\ y \in rcosets\ H$ **by** (*metis inv-closed rcosets-closed*)
with *x y HG U* **have** $(\lambda U{\in}rcosets\ H.\ U\ \#{>}\ inv\ x)\ ((\lambda U{\in}rcosets\ H.\ U\ \#{>}\ inv\ y)\ U) = U\ \#{>}\ inv\ y\ \#{>}\ inv\ x$ **by** *auto*
also from *x y U HG* **have** ... $= U\ \#{>}\ inv\ (x \otimes y)$ **by** (*metis inv-mult-group coset-mult-assoc inv-closed is-group subgroup.rcosets-carrier*)

finally show $(\lambda U{\in}rcosets\ H.\ U\ \#{>}\ inv\ x)\ ((\lambda U{\in}rcosets\ H.\ U\ \#{>}\ inv\ y)\ U) = U\ \#{>}\ inv\ (x \otimes y)$.
qed
finally show $(\lambda U{\in}rcosets\ H.\ U\ \#{>}\ inv\ (x \otimes y)) = (\lambda U{\in}rcosets\ H.\ U\ \#{>}\ inv\ x) \otimes_{BijGroup\ (rcosets\ H)} (\lambda U{\in}rcosets\ H.\ U\ \#{>}\ inv\ y)$ **unfolding** *cosx-def cosy-def* **by** *simp*
qed

end

theory *SubgroupConjugation*
imports *GroupAction*
begin

2 Conjugation of Subgroups and Cosets

This theory examines properties of the conjugation of subgroups of a fixed group as a group action

2.1 Definitions and Preliminaries

We define the set of all subgroups of G which have a certain cardinality. G will act on those sets. Afterwards some theorems which are already available for right cosets are dualized into statements about left cosets.

lemma (**in** *subgroup*) *subgroup-of-subset*:
assumes $G{:}group\ G$
assumes $PH{:}H \subseteq K$
assumes $KG{:}subgroup\ K\ G$
shows $subgroup\ H\ (G(\!|carrier := K|\!))$
using *assms subgroup-def group.subgroup-inv-equality m-inv-closed* **by** *fastforce*

context *group*
begin

definition $subgroups\text{-}of\text{-}size :: nat \Rightarrow$ -
where $subgroups\text{-}of\text{-}size\ p = \{H.\ subgroup\ H\ G \wedge card\ H = p\}$

lemma *lcosI*: $[|\ h \in H;\ H \subseteq carrier\ G;\ x \in carrier\ G|] ==> x \otimes h \in x <\#\ H$
by (*auto simp add*: *l-coset-def*)

lemma *lcoset-join2*:
assumes $H{:}subgroup\ H\ G$
assumes $g{:}g \in H$
shows $g <\#\ H = H$
proof *auto*
fix x

assume *x*:*x* ∈ *g* <# *H*
then obtain *h* **where** *h*:*h* ∈ *H* *x* = *g* ⊗ *h* **unfolding** *l-coset-def* **by** *auto*
with *g H* **show** *x* ∈ *H* **by** (*metis subgroup.m-closed*)
next
fix *x*
assume *x*:*x* ∈ *H*
with *g H* **have** *inv g* ⊗ *x* ∈ *H* **by** (*metis subgroup.m-closed subgroup.m-inv-closed*)
with *x g H* **show** *x* ∈ *g* <# *H* **by** (*metis is-group subgroup.lcos-module-rev subgroup.mem-carrier*)
qed

lemma *cardeq-rcoset*:
assumes *finite* (*carrier G*)
assumes *M* ⊆ *carrier G*
assumes *g* ∈ *carrier G*
shows *card* (*M* #> *g*) = *card M*
proof −
have *M* #> *g* ∈ *rcosets M* **by** (*metis assms*(*2*) *assms*(*3*) *rcosetsI*)
thus *card* (*M* #> *g*) = *card M* **by** (*metis assms*(*1*) *assms*(*2*) *card-cosets-equal*)
qed

lemma *cardeq-lcoset*:
assumes *finite* (*carrier G*)
assumes *M*:*M* ⊆ *carrier G*
assumes *g*:*g* ∈ *carrier G*
shows *card* (*g* <# *M*) = *card M*
proof −
have *bij-betw* (λ*m*. *g* ⊗ *m*) *M* (*g* <# *M*)
proof(*auto simp add*: *bij-betw-def*)
show *inj-on* (*op* ⊗ *g*) *M*
proof(*rule inj-onI*)
from *g* **have** *invg*:*inv g* ∈ *carrier G* **by** (*rule inv-closed*)
fix *x y*
assume *x*:*x* ∈ *M* **and** *y*:*y* ∈ *M*
with *M* **have** *xG*:*x* ∈ *carrier G* **and** *yG*:*y* ∈ *carrier G* **by** *auto*
assume *g* ⊗ *x* = *g* ⊗ *y*
hence (*inv g*) ⊗ (*g* ⊗ *x*) = (*inv g*) ⊗ (*g* ⊗ *y*) **by** *simp*
with *g invg xG yG* **have** (*inv g* ⊗ *g*) ⊗ *x* = (*inv g* ⊗ *g*) ⊗ *y* **by** (*metis m-assoc*)
with *g invg xG yG* **show** *x* = *y* **by** *simp*
qed
next
fix *x*
assume *x* ∈ *M*
thus *g* ⊗ *x* ∈ *g* <# *M* **unfolding** *l-coset-def* **by** *auto*
next
fix *x*
assume *x*:*x* ∈ *g* <# *M*
then obtain *m* **where** *x* = *g* ⊗ *m* *m* ∈ *M* **unfolding** *l-coset-def* **by** *auto*

thus x ∈ op ⊗ g ‘ M **by** simp
qed
thus card (g <# M) = card M **by** (metis bij-betw-same-card)
qed

2.2 Conjugation is a group action

We will now prove that conjugation acts on the subgroups of a certain group. A large part of this proof consists of showing that the conjugation of a subgroup with a group element is, again, a subgroup.

lemma conjugation-subgroup:
assumes HG:subgroup H G
assumes gG:g ∈ carrier G
shows subgroup (g <# (H #> inv g)) G
proof
from gG **have** inv g ∈ carrier G **by** (rule inv-closed)
with HG **have** (H #> inv g) ⊆ carrier G **by** (metis r-coset-subset-G subgroup-imp-subset)
with gG **show** g <# (H #> inv g) ⊆ carrier G **by** (metis l-coset-subset-G)
next
from gG **have** invgG:inv g ∈ carrier G **by** (metis inv-closed)
with HG **have** lcosSubset:(H #> inv g) ⊆ carrier G **by** (metis r-coset-subset-G subgroup-imp-subset)
fix x y
assume x:x ∈ g <# (H #> inv g) **and** y:y ∈ g <# (H #> inv g)
then obtain x′ y′ **where** x′:x′ ∈ H #> inv g x = g ⊗ x′ **and** y′:y′ ∈ H #> inv g y = g ⊗ y′ **unfolding** l-coset-def **by** auto
then obtain hx hy **where** hx:hx ∈ H x′ = hx ⊗ inv g **and** hy:hy ∈ H y′ = hy ⊗ inv g **unfolding** r-coset-def **by** auto
with x′ y′ **have** x2:x = g ⊗ (hx ⊗ inv g) **and** y2:y = g ⊗ (hy ⊗ inv g) **by** auto
hence x ⊗ y = (g ⊗ (hx ⊗ inv g)) ⊗ (g ⊗ (hy ⊗ inv g)) **by** simp
also from hx hy HG **have** hxG:hx ∈ carrier G **and** hyG:hy ∈ carrier G **by** (metis subgroup.mem-carrier)+
with gG hy x2 invgG **have** (g ⊗ (hx ⊗ inv g)) ⊗ (g ⊗ (hy ⊗ inv g)) = g ⊗ hx ⊗ (inv g ⊗ g) ⊗ hy ⊗ inv g **by** (metis m-assoc m-closed)
also from invgG gG **have** ... = g ⊗ hx ⊗ **1** ⊗ hy ⊗ inv g **by** simp
also from gG hxG **have** ... = g ⊗ hx ⊗ hy ⊗ inv g **by** (metis m-closed r-one)
also from gG hxG invgG **have** ... = g ⊗ ((hx ⊗ hy) ⊗ inv g) **by** (metis gG hxG hyG invgG m-assoc m-closed)
finally have xy:x ⊗ y = g ⊗ (hx ⊗ hy ⊗ inv g).
from hx hy HG **have** hx ⊗ hy ∈ H **by** (metis subgroup.m-closed)
with invgG HG **have** (hx ⊗ hy) ⊗ inv g ∈ H #> inv g **by** (metis rcosI subgroup-imp-subset)
with gG lcosSubset **have** g ⊗ (hx ⊗ hy ⊗ inv g) ∈ g <# (H #> inv g) **by** (metis lcosI)
with xy **show** x ⊗ y ∈ g <# (H #> inv g) **by** simp
next
from gG **have** invgG:inv g ∈ carrier G **by** (metis inv-closed)
with HG **have** lcosSubset:(H #> inv g) ⊆ carrier G **by** (metis r-coset-subset-G subgroup-imp-subset)

from *HG* **have** **1** ∈ *H* **by** (*rule subgroup.one-closed*)
with *invgG HG* **have** **1** ⊗ *inv g* ∈ *H* #> *inv g* **by** (*metis rcosI subgroup-imp-subset*)
with *gG lcosSubset* **have** *g* ⊗ (**1** ⊗ *inv g*) ∈ *g* <# (*H* #> *inv g*) **by** (*metis lcosI*)
with *gG invgG* **show** **1** ∈ *g* <# (*H* #> *inv g*) **by** *simp*
next
from *gG* **have** *invgG*:*inv g* ∈ *carrier G* **by** (*metis inv-closed*)
with *HG* **have** *lcosSubset*:(*H* #> *inv g*) ⊆ *carrier G* **by** (*metis r-coset-subset-G subgroup-imp-subset*)
fix *x*
assume *x* ∈ *g* <# (*H* #> *inv g*)
then obtain *x′* **where** *x′*:*x′* ∈ *H* #> *inv g x* = *g* ⊗ *x′* **unfolding** *l-coset-def* **by** *auto*
then obtain *hx* **where** *hx*:*hx* ∈ *H x′* = *hx* ⊗ *inv g* **unfolding** *r-coset-def* **by** *auto*
with *HG* **have** *invhx*:*inv hx* ∈ *H* **by** (*metis subgroup.m-inv-closed*)
from *x′ hx* **have** *inv x* = *inv* (*g* ⊗ (*hx* ⊗ *inv g*)) **by** *simp*
also from *x′ hx HG gG invgG* **have** ... = *inv* (*inv g*) ⊗ *inv hx* ⊗ *inv g* **by** (*metis calculation in-mono inv-mult-group lcosSubset subgroup.mem-carrier*)
also from *gG* **have** ... = *g* ⊗ *inv hx* ⊗ *inv g* **by** *simp*
also from *gG invgG invhx HG* **have** ... = *g* ⊗ (*inv hx* ⊗ *inv g*) **by** (*metis m-assoc subgroup.mem-carrier*)
finally have *invx*:*inv x* = *g* ⊗ (*inv hx* ⊗ *inv g*)**.**
with *invhx invgG HG* **have** (*inv hx*) ⊗ *inv g* ∈ *H* #> *inv g* **by** (*metis rcosI subgroup-imp-subset*)
with *gG lcosSubset* **have** *g* ⊗ (*inv hx* ⊗ *inv g*) ∈ *g* <# (*H* #> *inv g*) **by** (*metis lcosI*)
with *invx* **show** *inv x* ∈ *g* <# (*H* #> *inv g*) **by** *simp*
qed

definition *conjugation-action*::*nat* ⇒ -
where *conjugation-action p* = (λ*g*∈*carrier G*. λ*P*∈*subgroups-of-size p*. *g* <# (*P* #> *inv g*))

lemma *conjugation-is-size-invariant*:
assumes *fin*:*finite* (*carrier G*)
assumes *P*:*P* ∈ *subgroups-of-size p*
assumes *g*:*g* ∈ *carrier G*
shows *conjugation-action p g P* ∈ *subgroups-of-size p*
proof −
from *g* **have** *invg*:*inv g* ∈ *carrier G* **by** (*metis inv-closed*)
from *P* **have** *PG*:*subgroup P G* **and** *card*:*card P* = *p* **unfolding** *subgroups-of-size-def* **by** *simp+*
hence *PsubG*:*P* ⊆ *carrier G* **by** (*metis subgroup-imp-subset*)
hence *PinvgsubG*:*P* #> *inv g* ⊆ *carrier G* **by** (*metis invg r-coset-subset-G*)
have *g* <# (*P* #> *inv g*) ∈ *subgroups-of-size p*
proof(*auto simp add*:*subgroups-of-size-def*)
show *subgroup* (*g* <# (*P* #> *inv g*)) *G* **by** (*metis g PG conjugation-subgroup*)
next

from *card PsubG fin invg* **have** *card* (*P* #> *inv g*) = *p* **by** (*metis cardeq-rcoset*)
with *g PinvgsubG fin* **show** *card* (*g* <# (*P* #> *inv g*)) = *p* **by** (*metis cardeq-lcoset*)
qed
with *P g* **show** *?thesis* **unfolding** *conjugation-action-def* **by** *simp*
qed

lemma *conjugation-is-Bij*:
assumes *fin*:*finite* (*carrier G*)
assumes *g*:*g* ∈ *carrier G*
shows *conjugation-action p g* ∈ *Bij* (*subgroups-of-size p*)
proof −
from *g* **have** *invg*:*inv g* ∈ *carrier G* **by** (*rule inv-closed*)
from *g* **have** *conjugation-action p g* ∈ *extensional* (*subgroups-of-size p*) **unfolding** *conjugation-action-def* **by** *simp*
moreover have *bij-betw* (*conjugation-action p g*) (*subgroups-of-size p*) (*subgroups-of-size p*)
proof(*auto simp add*:*bij-betw-def*)
show *inj-on* (*conjugation-action p g*) (*subgroups-of-size p*)
proof(*rule inj-onI*)
fix *U V*
assume *U*:*U* ∈ *subgroups-of-size p* **and** *V*:*V* ∈ *subgroups-of-size p*
hence *subsetG*:*U* ⊆ *carrier G V* ⊆ *carrier G* **unfolding** *subgroups-of-size-def* **by** (*metis* (*lifting*) *mem-Collect-eq subgroup-imp-subset*)+
hence *subsetL*:*U* #> *inv g* ⊆ *carrier G V* #> *inv g* ⊆ *carrier G* **by** (*metis invg r-coset-subset-G*)+
assume *conjugation-action p g U* = *conjugation-action p g V*
with *g U V* **have** *g* <# (*U* #> *inv g*) = *g* <# (*V* #> *inv g*) **unfolding** *conjugation-action-def* **by** *simp*
hence (*inv g*) <# (*g* <# (*U* #> *inv g*)) = (*inv g*) <# (*g* <# (*V* #> *inv g*)) **by** *simp*
hence (*inv g* ⊗ *g*) <# (*U* #> *inv g*) = (*inv g* ⊗ *g*) <# (*V* #> *inv g*) **by** (*metis g invg lcos-m-assoc r-coset-subset-G subsetG*)
hence **1** <# (*U* #> *inv g*) = **1** <# (*V* #> *inv g*) **by** (*metis g l-inv*)
hence *U* #> *inv g* = *V* #> *inv g* **by** (*metis subsetL lcos-mult-one*)
hence (*U* #> *inv g*) #> *g* = (*V* #> *inv g*) #> *g* **by** *simp*
hence *U* #> (*inv g* ⊗ *g*) = *V* #> (*inv g* ⊗ *g*) **by** (*metis coset-mult-assoc g inv-closed subsetG*)
hence *U* #> **1** = *V* #> **1** **by** (*metis g l-inv*)
thus *U* = *V* **by** (*metis coset-mult-one subsetG*)
qed
next
fix *P*
assume *P* ∈ *subgroups-of-size p*
thus *conjugation-action p g P* ∈ *subgroups-of-size p* **by** (*metis fin g conjugation-is-size-invariant*)
next
fix *P*
assume *P*:*P* ∈ *subgroups-of-size p*
with *invg* **have** *conjugation-action p* (*inv g*) *P* ∈ *subgroups-of-size p* **by** (*metis*

fin invg conjugation-is-size-invariant)
with *invg P* **have** (*inv g*) <# (*P* #> (*inv* (*inv g*))) ∈ *subgroups-of-size p* **unfolding** *conjugation-action-def* **by** *simp*
hence *1*:(*inv g*) <# (*P* #> *g*) ∈ *subgroups-of-size p* **by** (*metis g inv-inv*)
have *g* <# (((*inv g*) <# (*P* #> *g*)) #> *inv g*) = (⋃*p* ∈ *P*. {*g* ⊗ (*inv g* ⊗ (*p* ⊗ *g*) ⊗ *inv g*)}) **unfolding** *r-coset-def l-coset-def* **by** (*simp add*:*m-assoc*)
also from *P* **have** *PG*:*P* ⊆ *carrier G* **unfolding** *subgroups-of-size-def* **by** (*auto simp add*:*subgroup-imp-subset*)
have ∀ *p* ∈ *P*. *g* ⊗ (*inv g* ⊗ (*p* ⊗ *g*) ⊗ *inv g*) = *p*
proof(*auto*)
fix *p*
assume *p* ∈ *P*
with *PG* **have** *p*:*p* ∈ *carrier G*..
with *g invg* **have** *g* ⊗ (*inv g* ⊗ (*p* ⊗ *g*) ⊗ *inv g*) = (*g* ⊗ *inv g*) ⊗ *p* ⊗ (*g* ⊗ *inv g*) **by** (*metis m-assoc m-closed*)
also with *g invg g p* **have** ... = *p* **by** (*metis l-one r-inv r-one*)
finally show *g* ⊗ (*inv g* ⊗ (*p* ⊗ *g*) ⊗ *inv g*) = *p*.
qed
hence (⋃*p* ∈ *P*. {*g* ⊗ (*inv g* ⊗ (*p* ⊗ *g*) ⊗ *inv g*)}) = *P* **by** *simp*
finally have *g* <# (((*inv g*) <# (*P* #> *g*)) #> *inv g*) = *P*.
with *1* **have** *P* ∈ (λ*P*. *g* <# (*P* #> *inv g*)) ‘ *subgroups-of-size p* **by** *auto*
with *P g* **show** *P* ∈ *conjugation-action p g* ‘ *subgroups-of-size p* **unfolding** *conjugation-action-def* **by** *simp*
qed
ultimately show *?thesis* **unfolding** *BijGroup-def Bij-def* **by** *simp*
qed

lemma *lr-coset-assoc*:
assumes *g*:*g* ∈ *carrier G*
assumes *h*:*h* ∈ *carrier G*
assumes *P*:*P* ⊆ *carrier G*
shows *g* <# (*P* #> *h*) = (*g* <# *P*) #> *h*
proof(*auto*)
fix *x*
assume *x* ∈ *g* <# (*P* #> *h*)
then obtain *p* **where** *p* ∈ *P* **and** *p*:*x* = *g* ⊗ (*p* ⊗ *h*) **unfolding** *l-coset-def r-coset-def* **by** *auto*
with *P* **have** *p* ∈ *carrier G* **by** *auto*
with *g h p* **have** *x* = (*g* ⊗ *p*) ⊗ *h* **by** (*metis m-assoc*)
with ‹*p* ∈ *P*› **show** *x* ∈ (*g* <# *P*) #> *h* **unfolding** *l-coset-def r-coset-def* **by** *auto*
next
fix *x*
assume *x* ∈ (*g* <# *P*) #> *h*
then obtain *p* **where** *p* ∈ *P* **and** *p*:*x* = (*g* ⊗ *p*) ⊗ *h* **unfolding** *l-coset-def r-coset-def* **by** *auto*
with *P* **have** *p* ∈ *carrier G* **by** *auto*
with *g h p* **have** *x* = *g* ⊗ (*p* ⊗ *h*) **by** (*metis m-assoc*)
with ‹*p* ∈ *P*› **show** *x* ∈ *g* <# (*P* #> *h*) **unfolding** *l-coset-def r-coset-def* **by**

auto
qed

theorem *acts-on-subsets*:
assumes *fin*:*finite* (*carrier G*)
shows *group-action G* (*conjugation-action p*) (*subgroups-of-size p*)
unfolding *group-action-def group-action-axioms-def group-hom-def group-hom-axioms-def hom-def*
apply(*auto simp add*:*is-group group-BijGroup*)
proof −
fix *g*
assume *g*:*g* ∈ *carrier G*
with *fin* **show** *conjugation-action p g* ∈ *carrier* (*BijGroup* (*subgroups-of-size p*))
unfolding *BijGroup-def* **by** (*metis conjugation-is-Bij partial-object.select-convs*(*1*))
next
fix *x y*
assume *x*:*x* ∈ *carrier G* **and** *y*:*y* ∈ *carrier G*
hence *invx*:*inv x* ∈ *carrier G* **and** *invy*:*inv y* ∈ *carrier G* **by** (*metis inv-closed*)+
from *x y* **have** *xyG*:*x* ⊗ *y* ∈ *carrier G* **by** (*metis m-closed*)
def *conjx* ≡ *conjugation-action p x*
def *conjy* ≡ *conjugation-action p y*
from *fin x* **have** *xBij*:*conjx* ∈ *Bij* (*subgroups-of-size p*) **unfolding** *conjx-def* **by** (*metis conjugation-is-Bij*)
from *fin y* **have** *yBij*:*conjy* ∈ *Bij* (*subgroups-of-size p*) **unfolding** *conjy-def* **by** (*metis conjugation-is-Bij*)
have *conjx* $\otimes_{BijGroup\ (subgroups\text{-}of\text{-}size\ p)}$ *conjy*
= (λ*g*∈*Bij* (*subgroups-of-size p*). *restrict* (*compose* (*subgroups-of-size p*) *g*) (*Bij* (*subgroups-of-size p*))) *conjx conjy* **unfolding** *BijGroup-def* **by** *simp*
also from *xBij yBij* **have** ... = *compose* (*subgroups-of-size p*) *conjx conjy* **by** *simp*
also have ... = (λ*P*∈*subgroups-of-size p*. *conjx* (*conjy P*)) **by** (*metis compose-def*)
also have ... = (λ*P*∈*subgroups-of-size p*. *x* ⊗ *y* <# (*P* #> *inv* (*x* ⊗ *y*)))
proof(*rule restrict-ext*)
fix *P*
assume *P*:*P* ∈ *subgroups-of-size p*
hence *PG*:*P* ⊆ *carrier G* **unfolding** *subgroups-of-size-def* **by** (*auto simp*:*subgroup-imp-subset*)
with *y* **have** *yPG*:*y* <# *P* ⊆ *carrier G* **by** (*metis l-coset-subset-G*)
from *x y* **have** *invxyG*:*inv* (*x* ⊗ *y*) ∈ *carrier G* **and** *xyG*:*x* ⊗ *y* ∈ *carrier G* **using** *inv-closed m-closed* **by** *auto*
from *yBij* **have** *conjy* ‘ *subgroups-of-size p* = *subgroups-of-size p* **unfolding** *Bij-def bij-betw-def* **by** *simp*
with *P* **have** *conjyP*:*conjy P* ∈ *subgroups-of-size p* **unfolding** *Bij-def bij-betw-def* **by** (*metis* (*full-types*) *imageI*)
with *x y P* **have** *conjx* (*conjy P*) = *x* <# ((*y* <# (*P* #> *inv y*)) #> *inv x*) **unfolding** *conjy-def conjx-def conjugation-action-def* **by** *simp*
also from *y invy PG* **have** ... = *x* <# (((*y* <# *P*) #> *inv y*) #> *inv x*) **by** (*metis lr-coset-assoc*)
also from *PG invx invy y* **have** ... = *x* <# ((*y* <# *P*) #> (*inv y* ⊗ *inv x*)) **by** (*metis coset-mult-assoc yPG*)

also from *x y* **have** ... = *x* <# ((*y* <# *P*) #> *inv* (*x* ⊗ *y*)) **by** (*metis inv-mult-group*)
also from *invxyG x yPG* **have** ... = (*x* <# (*y* <# *P*)) #> *inv* (*x* ⊗ *y*) **by** (*metis lr-coset-assoc*)
also from *x y PG* **have** ... = ((*x* ⊗ *y*) <# *P*) #> *inv* (*x* ⊗ *y*) **by** (*metis lcos-m-assoc*)
also from *xyG invxyG PG* **have** ... = (*x* ⊗ *y*) <# (*P* #> *inv* (*x* ⊗ *y*)) **by** (*metis lr-coset-assoc*)
finally show *conjx* (*conjy P*) = *x* ⊗ *y* <# (*P* #> *inv* (*x* ⊗ *y*))**.**
qed
finally have *conjx* $\otimes_{BijGroup\ (subgroups\text{-}of\text{-}size\ p)}$ *conjy* = (λ*P*∈*subgroups-of-size p*. *x* ⊗ *y* <# (*P* #> *inv* (*x* ⊗ *y*)))**.**
with *xyG* **show** *conjugation-action p* (*x* ⊗ *y*)
= *conjugation-action p x* $\otimes_{BijGroup\ (subgroups\text{-}of\text{-}size\ p)}$ *conjugation-action p y*
unfolding *conjx-def conjy-def conjugation-action-def* **by** *simp*
qed

2.3 Properties of the Conjugation Action

lemma *stabilizer-contains-P*:
assumes *fin*:*finite* (*carrier G*)
assumes *P*:*P* ∈ *subgroups-of-size p*
shows *P* ⊆ *group-action.stabilizer G* (*conjugation-action p*) *P*
proof
from *P* **have** *PG*:*subgroup P G* **unfolding** *subgroups-of-size-def* **by** *simp*
from *fin* **interpret** *conj*:*group-action G* (*conjugation-action p*) (*subgroups-of-size p*) **by** (*rule acts-on-subsets*)
fix *x*
assume *x*:*x* ∈ *P*
with *PG* **have** *inv x* ∈ *P* **by** (*metis subgroup.m-inv-closed*)
from *x P* **have** *xG*:*x* ∈ *carrier G* **unfolding** *subgroups-of-size-def subgroup-def* **by** *auto*
with *P* **have** *conjugation-action p x P* = *x* <# (*P* #> *inv x*) **unfolding** *conjugation-action-def* **by** *simp*
also from ‹*inv x* ∈ *P*› *PG* **have** ... = *x* <# *P* **by** (*metis coset-join2 subgroup.mem-carrier*)
also from *PG x* **have** ... = *P* **by** (*rule lcoset-join2*)
finally have *conjugation-action p x P* = *P***.**
with *xG* **show** *x* ∈ *group-action.stabilizer G* (*conjugation-action p*) *P* **unfolding** *conj.stabilizer-def* **by** *simp*
qed

corollary *stabilizer-supergrp-P*:
assumes *fin*:*finite* (*carrier G*)
assumes *P*:*P* ∈ *subgroups-of-size p*
shows *subgroup P* (*G*⦇*carrier* := *group-action.stabilizer G* (*conjugation-action p*) *P*⦈)
proof –
from *assms* **have** *P* ⊆ *group-action.stabilizer G* (*conjugation-action p*) *P* **by**

(*rule stabilizer-contains-P*)
moreover from *P* **have** *subgroup P G* **unfolding** *subgroups-of-size-def* **by** *simp*
moreover from *P fin* **have** *subgroup* (*group-action.stabilizer G* (*conjugation-action p*) *P*) *G* **by** (*metis acts-on-subsets group-action.stabilizer-is-subgroup*)
ultimately show *?thesis* **by** (*metis is-group subgroup.subgroup-of-subset*)
qed

lemma (**in** *group*) *P-fixed-point-of-P-conj*:
assumes *fin*:*finite* (*carrier G*)
assumes *P*:$P \in$ *subgroups-of-size p*
shows $P \in$ *group-action.fixed-points* (*G*(|*carrier* := *P*|)) (*conjugation-action p*) (*subgroups-of-size p*)
proof −
from *fin* **interpret** *conjG*: *group-action G conjugation-action p subgroups-of-size p* **by** (*rule acts-on-subsets*)
from *P* **have** *subgroup P G* **unfolding** *subgroups-of-size-def* **by** *simp*
with *fin* **interpret** *conjP*: *group-action G*(|*carrier* := *P*|) (*conjugation-action p*) (*subgroups-of-size p*) **by** (*metis acts-on-subsets group-action.subgroup-action*)
from *fin P* **have** $P \subseteq$ *conjG.stabilizer P* **by** (*rule stabilizer-contains-P*)
hence $P \subseteq$ *conjP.stabilizer P* **using** *conjG.stabilizer-def conjP.stabilizer-def* **by** *auto*
with *P* **show** $P \in$ *conjP.fixed-points* **unfolding** *conjP.fixed-points-def* **by** *auto*
qed

lemma *conj-wo-inv*:
assumes *QG*:*subgroup Q G*
assumes *PG*:*subgroup P G*
assumes *g*:$g \in$ *carrier G*
assumes *conj*:*inv g* <# (*Q* #> *g*) = *P*
shows *Q* #> *g* = *g* <# *P*
proof −
from *g* **have** *invg*:*inv g* $\in$ *carrier G* **by** (*metis inv-closed*)
from *conj* **have** *g* <# (*inv g* <# (*Q* #> *g*)) = *g* <# *P* **by** *simp*
with *QG g invg* **have** (*g* $\otimes$ *inv g*) <# (*Q* #> *g*) = *g* <# *P* **by** (*metis lcos-m-assoc r-coset-subset-G subgroup-imp-subset*)
with *g invg* **have** **1** <# (*Q* #> *g*) = *g* <# *P* **by** (*metis r-inv*)
with *QG g* **show** *Q* #> *g* = *g* <# *P* **by** (*metis lcos-mult-one r-coset-subset-G subgroup-imp-subset*)
qed

end

end

theory *SndSylow*
imports *SubgroupConjugation*
begin

3 The Secondary Sylow Theorems

3.1 Preliminaries

lemma *singletonI*:
  assumes $\bigwedge x.\ x \in A \Longrightarrow x = y$
  assumes $y \in A$
  shows $A = \{y\}$
using *assms* **by** *fastforce*

context *group*
begin

lemma *set-mult-inclusion*:
  assumes *H*:*subgroup H G*
  assumes *Q*:$P \subseteq$ *carrier G*
  assumes *PQ*:H <#> $P \subseteq H$
  shows $P \subseteq H$
proof
  fix x
  from H **have** $\mathbf{1} \in H$ **by** (*rule subgroup.one-closed*)
  moreover assume x:$x \in P$
  ultimately have $\mathbf{1} \otimes x \in H$ <#> P **unfolding** *set-mult-def* **by** *auto*
  with *PQ* **have** $\mathbf{1} \otimes x \in H$ **by** *auto*
  with *H Q x* **show** $x \in H$ **by** (*metis in-mono l-one*)
qed

lemma *card-subgrp-dvd*:
  assumes *subgroup H G*
  shows *card H dvd order G*
proof(*cases finite* (*carrier G*))
  case *True*
  with *assms* **have** *card* (*rcosets H*) $*$ *card H* = *order G* **by** (*metis lagrange*)
  thus *?thesis* **by** (*metis dvd-triv-left mult.commute*)
next
  case *False*
  hence *order G* = *0* **unfolding** *order-def* **by** (*metis card-infinite*)
  thus *?thesis* **by** (*metis dvd-0-right*)
qed

lemma *subgroup-finite*:
  assumes *subgroup*:*subgroup H G*
  assumes *finite*:*finite* (*carrier G*)
  shows *finite H*
by (*metis finite finite-subset subgroup subgroup-imp-subset*)

end

3.2 Extending the Sylow Locale

This locale extends the originale *sylow* locale by adding the constraint that the p must not divide the remainder m, i.e. p^a is the maximal size of a p-subgroup of G.

locale *snd-sylow* = *sylow* +
assumes *pNotDvdm*:¬ (*p dvd m*)

context *snd-sylow*
begin

lemma *pa-not-zero*: *p* ^ *a* ≠ *0*
by (*simp add*: *prime-gt-0-nat prime-p*)

lemma *sylow-greater-zero*:
shows *card* (*subgroups-of-size* (*p* ^ *a*)) > *0*
proof −
obtain *P* **where** *PG*:*subgroup P G* **and** *cardP*:*card P* = *p* ^ *a* **by** (*metis sylow-thm*)
hence *P* ∈ *subgroups-of-size* (*p* ^ *a*) **unfolding** *subgroups-of-size-def* **by** *auto*
hence *subgroups-of-size* (*p* ^ *a*) ≠ {} **by** *auto*
moreover from *finite-G* **have** *finite* (*subgroups-of-size* (*p* ^ *a*)) **unfolding** *subgroups-of-size-def subgroup-def* **by** *auto*
ultimately show *?thesis* **by** *auto*
qed

lemma *is-snd-sylow*: *snd-sylow G p a m* **by** (*rule snd-sylow-axioms*)

3.3 Every p-group is Contained in a conjugate of a p-Sylow-Group

lemma *ex-conj-sylow-group*:
assumes *H*:*H* ∈ *subgroups-of-size* (*p* ^ *b*)
assumes *Psize*:*P* ∈ *subgroups-of-size* (*p* ^ *a*)
obtains *g* **where** *g* ∈ *carrier G H* ⊆ *g* <# (*P* #> *inv g*)
proof −
from *H* **have** *HsubG*:*subgroup H G* **unfolding** *subgroups-of-size-def* **by** *auto*
hence *HG*:*H* ⊆ *carrier G* **unfolding** *subgroups-of-size-def* **by** (*simp add*:*subgroup-imp-subset*)
from *Psize* **have** *PG*:*subgroup P G* **and** *cardP*:*card P* = *p* ^ *a* **unfolding** *subgroups-of-size-def* **by** *auto*
def *H'* ≡ *G*(|*carrier* := *H*|)
from *HsubG* **interpret** *Hgroup*: *group H'* **unfolding** *H'-def* **by** (*metis subgroup-imp-group*)
from *H* **have** *orderH'*:*order H'* = *p* ^ *b* **unfolding** *H'-def subgroups-of-size-def order-def* **by** *simp*
def *φ* ≡ *λg*. *λU*∈*rcosets P*. *U* #> *inv g*
with *PG* **interpret** *Gact*: *group-action G φ rcosets P* **unfolding** *φ-def* **by** (*metis inv-mult-on-rcosets-action*)
from *H* **interpret** *H'act*: *group-action H' φ rcosets P* **unfolding** *H'-def subgroups-of-size-def* **by** (*metis* (*mono-tags*) *Gact.subgroup-action mem-Collect-eq*)

from finite-G PG **have** finite (rcosets P) **unfolding** RCOSETS-def r-coset-def **by** (metis (lifting) finite.emptyI finite-UN-I finite-insert)
with orderH' sylow-axioms cardP **have** card H'act.fixed-points mod p = card (rcosets P) mod p **unfolding** sylow-def sylow-axioms-def **by** (metis H'act.fixed-point-congruence)
moreover from finite-G PG order-G cardP **have** card (rcosets P) ∗ p ˆ a = p ˆ a ∗ m **by** (metis lagrange)
with prime-p **have** card (rcosets P) = m **by** (metis less-nat-zero-code mult-cancel2 mult-is-0 mult.commute order-G zero-less-o-G)
hence card (rcosets P) mod p = m mod p **by** simp
moreover from pNotDvdm prime-p **have** ... ≠ 0 **by** (metis dvd-eq-mod-eq-0)
ultimately have card H'act.fixed-points ≠ 0 **by** (metis mod-0)
then obtain N **where** N:N ∈ H'act.fixed-points **by** fastforce
hence Ncoset:N ∈ rcosets P **unfolding** H'act.fixed-points-def **by** simp
then obtain g **where** g:g ∈ carrier G N = P #> g **unfolding** RCOSETS-def **by** auto
hence invg:inv g ∈ carrier G **by** (metis inv-closed)
hence invinvg:inv (inv g) ∈ carrier G **by** (metis inv-closed)
from N **have** carrier H' ⊆ H'act.stabilizer N **unfolding** H'act.fixed-points-def **by** simp
hence ∀ h∈H. φ h N = N **unfolding** H'act.stabilizer-def **using** H'-def **by** auto
with HG Ncoset **have** a1:∀ h∈H. N #> inv h ⊆ N **unfolding** φ-def **by** simp
have N <#> H ⊆ N **unfolding** set-mult-def r-coset-def
proof(auto)
  fix n h
  assume n:n ∈ N **and** h:h ∈ H
  with H **have** inv h ∈ H **by** (metis (mono-tags) mem-Collect-eq subgroup.m-inv-closed subgroups-of-size-def)
  with n HG PG a1 **have** n ⊗ inv (inv h) ∈ N **unfolding** r-coset-def **by** auto
  with HG h **show** n ⊗ h ∈ N **by** (metis in-mono inv-inv)
qed
with g **have** ((P #> g) <#> H) #> inv g ⊆ (P #> g) #> inv g **unfolding** r-coset-def **by** auto
with PG g invg **have** ((P #> g) <#> H) #> inv g ⊆ P **by** (metis coset-mult-assoc coset-mult-one r-inv subgroup-imp-subset)
with g HG PG invg **have** P <#> (g <# H #> inv g) ⊆ P **by** (metis lr-coset-assoc r-coset-subset-G rcos-assoc-lcos setmult-rcos-assoc subgroup-imp-subset)
with PG HG g invg **have** g <# H #> inv g ⊆ P **by** (metis l-coset-subset-G r-coset-subset-G set-mult-inclusion)
with g **have** (g <# H #> inv g) #> inv (inv g) ⊆ P #> inv (inv g) **unfolding** r-coset-def **by** auto
with HG g invg invinvg **have** g <# H ⊆ P #> inv (inv g) **by** (metis coset-mult-assoc coset-mult-inv2 l-coset-subset-G)
with g **have** (inv g) <# (g <# H) ⊆ inv g <# (P #> inv (inv g)) **unfolding** l-coset-def **by** auto
with HG g invg invinvg **have** H ⊆ inv g <# (P #> inv (inv g)) **by** (metis inv-inv lcos-m-assoc lcos-mult-one r-inv)
with invg **show** thesis **by** (auto dest:that)
qed

3.4 Every p-Group is Contained in a p-Sylow-Group

theorem *sylow-contained-in-sylow-group*:
  assumes *H*:*H* ∈ *subgroups-of-size* (*p* ^ *b*)
  obtains *S* **where** *H* ⊆ *S* **and** *S* ∈ *subgroups-of-size* (*p* ^ *a*)
proof −
  from *H* **have** *HG*:*H* ⊆ *carrier G* **unfolding** *subgroups-of-size-def* **by** (*simp add*:*subgroup-imp-subset*)
  obtain *P* **where** *PG*:*subgroup P G* **and** *cardP*:*card P* = *p* ^ *a* **by** (*metis sylow-thm*)
  hence *Psize*:*P* ∈ *subgroups-of-size* (*p* ^ *a*) **unfolding** *subgroups-of-size-def* **by** *simp*
  with *H* **obtain** *g* **where** *g*:*g* ∈ *carrier G H* ⊆ *g* <# (*P* #> *inv g*) **by** (*metis ex-conj-sylow-group*)
  moreover note *Psize g*
  moreover with *finite-G* **have** *conjugation-action* (*p* ^ *a*) *g P* ∈ *subgroups-of-size* (*p* ^ *a*) **by** (*metis conjugation-is-size-invariant*)
  ultimately show *thesis* **unfolding** *conjugation-action-def* **by** (*auto dest*:*that*)
qed

3.5 p-Sylow-Groups are conjugates of each other

theorem *sylow-conjugate*:
  assumes *P*:*P* ∈ *subgroups-of-size* (*p* ^ *a*)
  assumes *Q*:*Q* ∈ *subgroups-of-size* (*p* ^ *a*)
  obtains *g* **where** *g* ∈ *carrier G Q* = *g* <# (*P* #> *inv g*)
proof −
  from *P* **have** *card P* = *p* ^ *a* **unfolding** *subgroups-of-size-def* **by** *simp*
  from *Q* **have** *Qcard*:*card Q* = *p* ^ *a* **unfolding** *subgroups-of-size-def* **by** *simp*
  from *Q P* **obtain** *g* **where** *g*:*g* ∈ *carrier G Q* ⊆ *g* <# (*P* #> *inv g*) **by** (*rule ex-conj-sylow-group*)
  moreover with *P finite-G* **have** *conjugation-action* (*p* ^ *a*) *g P* ∈ *subgroups-of-size* (*p* ^ *a*) **by** (*metis conjugation-is-size-invariant*)
  moreover from *g P* **have** *conjugation-action* (*p* ^ *a*) *g P* = *g* <# (*P* #> *inv g*) **unfolding** *conjugation-action-def* **by** *simp*
  ultimately have *conjSize*:*g* <# (*P* #> *inv g*) ∈ *subgroups-of-size* (*p* ^ *a*) **unfolding** *conjugation-action-def* **by** *simp*
  with *Qcard* **have** *card*:*card* (*g* <# (*P* #> *inv g*)) = *card Q* **unfolding** *subgroups-of-size-def* **by** *simp*
  from *conjSize finite-G* **have** *finite* (*g* <# (*P* #> *inv g*)) **by** (*metis* (*mono-tags*) *finite-subset mem-Collect-eq subgroup-imp-subset subgroups-of-size-def*)
  with *g card* **have** *Q* = *g* <# (*P* #> *inv g*) **by** (*metis card-subset-eq*)
  with *g* **show** *thesis* **by** (*metis that*)
qed

corollary *sylow-conj-orbit-rel*:
  assumes *P*:*P* ∈ *subgroups-of-size* (*p* ^ *a*)
  assumes *Q*:*Q* ∈ *subgroups-of-size* (*p* ^ *a*)
  shows (*P*,*Q*) ∈ *group-action.same-orbit-rel G* (*conjugation-action* (*p* ^ *a*)) (*subgroups-of-size* (*p* ^ *a*))

unfolding *group-action.same-orbit-rel-def*
proof −
  from *Q P* **obtain** *g* **where** *g*:*g* ∈ *carrier G P* = *g* <# (*Q* #> *inv g*) **by** (*rule sylow-conjugate*)
  with *Q P* **have** *g'*:*conjugation-action* (*p* ^ *a*) *g Q* = *P* **unfolding** *conjugation-action-def* **by** *simp*
  from *finite-G* **interpret** *conj*: *group-action G* (*conjugation-action* (*p* ^ *a*)) (*subgroups-of-size* (*p* ^ *a*)) **by** (*rule acts-on-subsets*)
  have *conj.same-orbit-rel* = {*X* ∈ (*subgroups-of-size* (*p* ^ *a*) × *subgroups-of-size* (*p* ^ *a*)). ∃ *g* ∈ *carrier G*. ((*conjugation-action* (*p* ^ *a*)) *g*) (*snd X*) = (*fst X*)} **by** (*rule conj.same-orbit-rel-def*)
  with *g g' P Q* **show** *?thesis* **by** *auto*
qed

3.6 Counting Sylow-Groups

The number of sylow groups is the orbit size of one of them:

theorem *num-eq-card-orbit*:
  assumes *P*:*P* ∈ *subgroups-of-size* (*p* ^ *a*)
  shows *subgroups-of-size* (*p* ^ *a*) = *group-action.orbit G* (*conjugation-action* (*p* ^ *a*)) (*subgroups-of-size* (*p* ^ *a*)) *P*
proof(*auto*)
  from *finite-G* **interpret** *conj*: *group-action G* (*conjugation-action* (*p* ^ *a*)) (*subgroups-of-size* (*p* ^ *a*)) **by** (*rule acts-on-subsets*)
  have *group-action.orbit G* (*conjugation-action* (*p* ^ *a*)) (*subgroups-of-size* (*p* ^ *a*)) *P* = *group-action.same-orbit-rel G* (*conjugation-action* (*p* ^ *a*)) (*subgroups-of-size* (*p* ^ *a*)) `` {*P*} **by** (*rule conj.orbit-def*)
  fix *Q*
  {
    assume *Q*:*Q* ∈ *subgroups-of-size* (*p* ^ *a*)
    from *P Q* **obtain** *g* **where** *g*:*g* ∈ *carrier G Q* = *g* <# (*P* #> *inv g*) **by** (*rule sylow-conjugate*)
    with *P conj.orbit-char* **show** *Q* ∈ *group-action.orbit G* (*conjugation-action* (*p* ^ *a*)) (*subgroups-of-size* (*p* ^ *a*)) *P*
      unfolding *conjugation-action-def* **by** *auto*
  } {
    assume *Q* ∈ *group-action.orbit G* (*conjugation-action* (*p* ^ *a*)) (*subgroups-of-size* (*p* ^ *a*)) *P*
    with *P conj.orbit-char* **obtain** *g* **where** *g*:*g* ∈ *carrier G Q* = *conjugation-action* (*p* ^ *a*) *g P* **by** *auto*
    with *finite-G P* **show** *Q* ∈ *subgroups-of-size* (*p* ^ *a*) **by** (*metis conjugation-is-size-invariant*)
  }
qed

theorem *num-sylow-normalizer*:
  assumes *Psize*:*P* ∈ *subgroups-of-size* (*p* ^ *a*)
  shows *card* (*rcosets*$_{G(\!| carrier := group\text{-}action.stabilizer\ G\ (conjugation\text{-}action\ (p\ \hat{}\ a))\ P |\!)}$ *P*) ∗ *p* ^ *a* = *card* (*group-action.stabilizer G* (*conjugation-action* (*p* ^ *a*)) *P*)
proof −

from *finite-G* **interpret** *conj*: *group-action G* (*conjugation-action* (*p* ^ *a*)) (*subgroups-of-size* (*p* ^ *a*)) **by** (*rule acts-on-subsets*)
from *Psize* **have** *PG*:*subgroup P G* **and** *cardP*:*card P* = *p* ^ *a* **unfolding** *subgroups-of-size-def* **by** *auto*
with *finite-G* **have** *order G* = *card* (*conj.orbit P*) ∗ *card* (*conj.stabilizer P*) **by** (*metis Psize acts-on-subsets group-action.orbit-size*)
with *order-G Psize* **have** *p* ^ *a* ∗ *m* = *card* (*subgroups-of-size* (*p* ^ *a*)) ∗ *card* (*conj.stabilizer P*) **by** (*metis num-eq-card-orbit*)
moreover from *Psize* **interpret** *stabGroup*: *group G*(|*carrier* := *conj.stabilizer P*|) **by** (*metis conj.stabilizer-is-subgroup subgroup-imp-group*)
from *finite-G Psize* **have** *PStab*:*subgroup P* (*G*(|*carrier* := *conj.stabilizer P*|)) **by** (*rule stabilizer-supergrp-P*)
from *finite-G Psize* **have** *finite* (*conj.stabilizer P*) **by** (*metis card-infinite conj.stabilizer-is-subgroup less-nat-zero-code subgroup.finite-imp-card-positive*)
with *finite-G PStab stabGroup.lagrange* **have** *card* (*rcosets*$_{G(|carrier\ :=\ conj.stabilizer\ P|)}$ *P*) ∗ *card P* = *order* (*G*(|*carrier* := *conj.stabilizer P*|)) **by** *force*
with *cardP* **show** *?thesis* **unfolding** *order-def* **by** *auto*
qed

theorem (**in** *snd-sylow*) *num-sylow-dvd-remainder*:
shows *card* (*subgroups-of-size* (*p* ^ *a*)) *dvd m*
proof −
from *finite-G* **interpret** *conj*: *group-action G* (*conjugation-action* (*p* ^ *a*)) (*subgroups-of-size* (*p* ^ *a*)) **by** (*rule acts-on-subsets*)
obtain *P* **where** *PG*:*subgroup P G* **and** *cardP*:*card P* = *p* ^ *a* **by** (*metis sylow-thm*)
hence *Psize*:*P* ∈ *subgroups-of-size* (*p* ^ *a*) **unfolding** *subgroups-of-size-def* **by** *simp*
with *finite-G* **have** *order G* = *card* (*conj.orbit P*) ∗ *card* (*conj.stabilizer P*) **by** (*metis Psize acts-on-subsets group-action.orbit-size*)
with *order-G Psize* **have** *orderEq*:*p* ^ *a* ∗ *m* = *card* (*subgroups-of-size* (*p* ^ *a*)) ∗ *card* (*conj.stabilizer P*) **by** (*metis num-eq-card-orbit*)
def *k* ≡ *card* (*rcosets*$_{G(|carrier\ :=\ conj.stabilizer\ P|)}$ *P*)
with *Psize* **have** *k* ∗ *p* ^ *a* = *card* (*conj.stabilizer P*) **by** (*metis num-sylow-normalizer*)
with *orderEq* **have** *p* ^ *a* ∗ *m* = *card* (*subgroups-of-size* (*p* ^ *a*)) ∗ *p* ^ *a* ∗ *k* **by** (*auto simp*:*mult.assoc mult.commute*)
hence *p* ^ *a* ∗ *m* = *p* ^ *a* ∗ *card* (*subgroups-of-size* (*p* ^ *a*)) ∗ *k* **by** *auto*
then have *m* = *card* (*subgroups-of-size* (*p* ^ *a*)) ∗ *k*
using *pa-not-zero* **by** *auto*
then show *?thesis* **..**
qed

We can restrict this locale to refer to a subgroup of order at least p^a:

lemma (**in** *snd-sylow*) *restrict-locale*:
assumes *subgrp*:*subgroup P G*
assumes *card*:*p* ^ *a dvd card P*
shows *snd-sylow* (*G*(|*carrier* := *P*|)) *p a* ((*card P*) *div* (*p* ^ *a*))
proof −
from *subgrp* **interpret** *groupP*: *group G*(|*carrier* := *P*|) **by** (*metis subgroup-imp-group*)

def *k* ≡ (*card P*) *div* (*p* ˆ *a*)
with *card* **have** *cardP*:*card P* = *p* ˆ *a* ∗ *k* **by** (*auto simp*: *dvd-mult-div-cancel*)
hence *orderP*:*order* (*G*(|*carrier* := *P*|)) = *p* ˆ *a* ∗ *k* **unfolding** *order-def* **by** *simp*
from *cardP subgrp order-G* **have** *p* ˆ *a* ∗ *k dvd p* ˆ *a* ∗ *m* **by** (*metis card-subgrp-dvd*)
hence *k dvd m*
by (*metis nat-mult-dvd-cancel-disj pa-not-zero*)
with *pNotDvdm* **have** *ndvd*:¬ *p dvd k*
by (*blast intro*: *dvd-trans*)
def *PcalM* ≡ {*s*. *s* ⊆ *carrier* (*G*(|*carrier* := *P*|)) ∧ *card s* = *p* ˆ *a*}
def *PRelM* ≡ {(*N1*, *N2*). *N1* ∈ *PcalM* ∧ *N2* ∈ *PcalM* ∧ (∃ *g*∈*carrier* (*G*(|*carrier* := *P*|)). *N1* = *N2* #>$_{G(|carrier\ :=\ P|)}$ *g*)}
from *subgrp finite-G* **have** *finite-groupP*:*finite* (*carrier* (*G*(|*carrier* := *P*|))) **by** (*auto simp*:*subgroup-finite*)
interpret *Nsylow*: *snd-sylow G*(|*carrier* := *P*|) *p a k PcalM PRelM*
unfolding *snd-sylow-def snd-sylow-axioms-def sylow-def sylow-axioms-def k-def*
using *groupP.is-group prime-p orderP finite-groupP ndvd PcalM-def PRelM-def k-def* **by** *fastforce+*
show *?thesis* **using** *k-def* **by** (*metis Nsylow.is-snd-sylow*)
qed

theorem (**in** *snd-sylow*) *p-sylow-mod-p*:
shows *card* (*subgroups-of-size* (*p* ˆ *a*)) *mod p* = *1*
proof −
obtain *P* **where** *PG*:*subgroup P G* **and** *cardP*:*card P* = *p* ˆ *a* **by** (*metis sylow-thm*)
hence *orderP*:*order* (*G*(|*carrier* := *P*|)) = *p* ˆ *a* **unfolding** *order-def* **by** *auto*
from *PG* **have** *PsubG*:*P* ⊆ *carrier G* **by** (*metis subgroup-imp-subset*)
from *PG cardP* **have** *PSize*:*P* ∈ *subgroups-of-size* (*p* ˆ *a*) **unfolding** *subgroups-of-size-def* **by** *auto*
from *PG* **interpret** *groupP*:*group* (*G*(|*carrier* := *P*|)) **by** (*rule subgroup-imp-group*)
from *cardP* **have** *PSize2*:*P* ∈ *groupP.subgroups-of-size* (*p* ˆ *a*) **using** *groupP.subgroups-of-size-def groupP.subgroup-self* **by** *auto*
from *finite-G* **interpret** *conjG*: *group-action G conjugation-action* (*p* ˆ *a*) *subgroups-of-size* (*p* ˆ *a*) **by** (*rule acts-on-subsets*)
from *PG* **interpret** *conjP*: *group-action G*(|*carrier* := *P*|) *conjugation-action* (*p* ˆ *a*) *subgroups-of-size* (*p* ˆ *a*) **by** (*rule conjG.subgroup-action*)
from *finite-G* **have** *finite* (*subgroups-of-size* (*p* ˆ *a*)) **unfolding** *subgroups-of-size-def subgroup-def* **by** *auto*
with *orderP prime-p* **have** *card* (*subgroups-of-size* (*p* ˆ *a*)) *mod p* = *card conjP.fixed-points mod p* **by** (*rule conjP.fixed-point-congruence*)
also have ... = *1*
proof −
have ⋀*Q*. *Q* ∈ *conjP.fixed-points* ⟹ *Q* = *P*
proof −
fix *Q*
assume *Qfixed*:*Q* ∈ *conjP.fixed-points*
hence *Qsize*:*Q* ∈ *subgroups-of-size* (*p* ˆ *a*) **unfolding** *conjP.fixed-points-def* **by** *simp*

hence *cardQ*:*card* $Q = p$ ^ a **unfolding** *subgroups-of-size-def* **by** *simp*
— The normalizer of Q in G
— Let's first show some basic propertiers of N
def $N \equiv$ *conjG.stabilizer* Q
def $k \equiv$ (*card* N) *div* (p ^ a)
from *N-def Qsize* **have** *NG*:*subgroup* N G **by** (*metis conjG.stabilizer-is-subgroup*)
then interpret *groupN*: *group* $G(|carrier := N|)$ **by** (*metis subgroup-imp-group*)
from *Qsize N-def* **have** *QN*:*subgroup* Q ($G(|carrier := N|)$) **using** *stabilizer-supergrp-P* **by** *auto*
— The following proposition is used to show that $P = Q$ later
from *Qsize* **have** *NfixesQ*:$\forall g \in N$. *conjugation-action* (p ^ a) g $Q = Q$ **unfolding** *N-def conjG.stabilizer-def* **by** *auto*
from *Qfixed* **have** *PfixesQ*:$\forall g \in P$. *conjugation-action* (p ^ a) g $Q = Q$ **unfolding** *conjP.fixed-points-def conjP.stabilizer-def* **by** *auto*
with *PsubG* **have** $P \subseteq N$ **unfolding** *N-def conjG.stabilizer-def* **by** *auto*
with *PG N-def Qsize* **have** *PN*:*subgroup* P ($G(|carrier := N|)$) **by** (*metis conjG.stabilizer-is-subgroup is-group subgroup.subgroup-of-subset*)
with *cardP* **have** p ^ a *dvd order* ($G(|carrier := N|)$) **using** *groupN.card-subgrp-dvd* **by** *force*
hence p ^ a *dvd card* N **unfolding** *order-def* **by** *simp*
with *NG* **have** *smaller-sylow*:*snd-sylow* ($G(|carrier := N|)$) p a k **unfolding** *k-def* **by** (*rule restrict-locale*)
— Instantiate the *snd-sylow* Locale a second time for the normalizer of Q
def *NcalM* $\equiv \{s.\ s \subseteq$ *carrier* ($G(|carrier := N|)$) $\wedge$ *card* $s = p$ ^ $a\}$
def *NRelM* $\equiv \{(N1,\ N2).\ N1 \in NcalM \wedge N2 \in NcalM \wedge (\exists g \in carrier\ (G(|carrier := N|)).\ N1 = N2 \#>_{G(|carrier := N|)} g)\}$
interpret *Nsylow*: *snd-sylow* $G(|carrier := N|)$ p a k *NcalM NRelM* **using** *smaller-sylow NcalM-def NRelM-def*.
— P and Q are conjugate in N:
from *cardP PN* **have** *PsizeN*:$P \in$ *groupN.subgroups-of-size* (p ^ a) **unfolding** *groupN.subgroups-of-size-def* **by** *auto*
from *cardQ QN* **have** *QsizeN*:$Q \in$ *groupN.subgroups-of-size* (p ^ a) **unfolding** *groupN.subgroups-of-size-def* **by** *auto*
from *QsizeN PsizeN* **obtain** g **where** g:$g \in$ *carrier* ($G(|carrier := N|)$) $P = g <\#_{G(|carrier := N|)} (Q \#>_{G(|carrier := N|)} inv_{G(|carrier := N|)}\ g)$ **by** (*rule Nsylow.sylow-conjugate*)
with *NG* **have** $P = g <\# (Q \#> inv\ g)$ **unfolding** *r-coset-def l-coset-def* **by** (*auto simp*:*subgroup-inv-equality*)
with *NG g Qsize* **have** *conjugation-action* (p ^ a) g $Q = P$ **unfolding** *conjugation-action-def* **using** *subgroup-imp-subset* **by** *force*
with *g NfixesQ* **show** $Q = P$ **by** *auto*
qed
moreover from *finite-G PSize* **have** $P \in$ *conjP.fixed-points* **using** *P-fixed-point-of-P-conj* **by** *auto*
ultimately have *conjP.fixed-points* $= \{P\}$ **by** *fastforce*
hence *one*:*card conjP.fixed-points* $= 1$ **by** (*auto simp*: *card-Suc-eq*)
with *prime-p* **have** *card conjP.fixed-points* $< p$ **unfolding** *prime-nat-iff* **by** *auto*
with *one* **show** *?thesis* **using** *mod-pos-pos-trivial* **by** *auto*

  qed
  finally show *?thesis***.**
qed

end

end