Topological Groups

Niklas Krofta

March 17, 2025

Abstract

Topological groups are blends of groups and topological spaces with the property that the multiplication and inversion operations are continuous functions. They frequently occur in mathematics and physics, e.g. in the form of Lie groups. We formalize the theory of topological groups on top of HOL-Algebra and HOL-Analysis. Topological groups are defined via a locale. We also introduce a set-based notion of uniform spaces in order to define the uniform structures of topological groups. The most notable formalized result is the Birkhoff-Kakutani theorem which characterizes metrizable topological groups. Our formalization also defines the important matrix groups $\mathrm{GL}_n(\mathbb{R})$, $\mathrm{SL}_n(\mathbb{R})$, O_n, SO_n and proves them to be topological groups.

The formalized results and proofs have been taken from the textbooks of Arhangelskii and Tkachenko [1], Bump [2] and James [4]. These lecture notes [5] have also been helpful.

Contents

1 Uniform spaces

theory *Uniform-Structure*
imports *HOL−Analysis.Abstract-Topology HOL−Analysis.Abstract-Metric-Spaces*
begin

Summary This section introduces a set-based notion of uniformities and connects it to the *uniform-space* type class.

1.1 Definitions and basic results

definition *uniformity-on* :: $'a\ set \Rightarrow (('a \times 'a)\ set \Rightarrow bool) \Rightarrow bool$ **where**
uniformity-on $X\ \mathcal{E} \longleftrightarrow$
$(\exists E.\ \mathcal{E}\ E) \wedge$
$(\forall E.\ \mathcal{E}\ E \longrightarrow E \subseteq X \times X \wedge Id\text{-}on\ X \subseteq E \wedge \mathcal{E}\ (E^{-1}) \wedge (\exists F.\ \mathcal{E}\ F \wedge F\ O\ F \subseteq E) \wedge$
$(\forall F.\ E \subseteq F \wedge F \subseteq X \times X \longrightarrow \mathcal{E}\ F)) \wedge$
$(\forall E\ F.\ \mathcal{E}\ E \longrightarrow \mathcal{E}\ F \longrightarrow \mathcal{E}\ (E \cap F))$

typedef $'a$ *uniformity* = $\{(X :: 'a\ set, \mathcal{E}).\ uniformity\text{-}on\ X\ \mathcal{E}\}$
morphisms *uniformity-rep uniformity*
proof −
have *uniformity-on UNIV* $(\lambda E.\ E = UNIV \times UNIV)$
unfolding *uniformity-on-def Id-on-def relcomp-def* **by** *auto*
then show *?thesis* **by** *fast*
qed

definition *uspace* :: $'a\ uniformity \Rightarrow 'a\ set$ **where**
uspace $\Phi = (let\ (X, \mathcal{E}) = uniformity\text{-}rep\ \Phi\ in\ X)$

definition *entourage-in* :: $'a\ uniformity \Rightarrow ('a \times 'a)\ set \Rightarrow bool$ **where**
entourage-in $\Phi = (let\ (X, \mathcal{E}) = uniformity\text{-}rep\ \Phi\ in\ \mathcal{E})$

lemma *uniformity-inverse'*[*simp*]:
assumes *uniformity-on* $X\ \mathcal{E}$
shows *uspace* $(uniformity\ (X, \mathcal{E})) = X \wedge entourage\text{-}in\ (uniformity\ (X, \mathcal{E})) = \mathcal{E}$
proof −

from *assms* **have** *uniformity-rep* (*uniformity* $(X, \mathcal{E})) = (X, \mathcal{E})$
  using *uniformity-inverse* **by** *blast*
then show *?thesis* **by** (*auto simp*: *prod.splits uspace-def entourage-in-def*)
qed

lemma *uniformity-entourages*:
shows *uniformity-on* (*uspace* Φ) (*entourage-in* Φ)
by (*metis Product-Type.Collect-case-prodD entourage-in-def split-beta uspace-def uniformity-rep*)

lemma *entourages-exist*: $\exists E.$ *entourage-in* Φ E
using *uniformity-entourages* **unfolding** *uniformity-on-def* **by** *blast*

lemma *entourage-in-space*[*elim*]: *entourage-in* Φ $E \Longrightarrow E \subseteq$ *uspace* $\Phi \times$ *uspace* Φ
using *uniformity-entourages* **unfolding** *uniformity-on-def* **by** *metis*

lemma *entourage-superset*[*intro*]:
entourage-in Φ $E \Longrightarrow E \subseteq F \Longrightarrow F \subseteq$ *uspace* $\Phi \times$ *uspace* $\Phi \Longrightarrow$ *entourage-in* Φ F
using *uniformity-entourages* **unfolding** *uniformity-on-def* **by** *blast*

lemma *entourage-intersection*[*intro*]: *entourage-in* Φ $E \Longrightarrow$ *entourage-in* Φ $F \Longrightarrow$ *entourage-in* Φ $(E \cap F)$
using *uniformity-entourages* **unfolding** *uniformity-on-def* **by** *metis*

lemma *entourage-converse*[*intro*]: *entourage-in* Φ $E \Longrightarrow$ *entourage-in* Φ (E^{-1})
using *uniformity-entourages* **unfolding** *uniformity-on-def* **by** *fast*

lemma *entourage-diagonal*[*dest*]:
assumes *entourage*: *entourage-in* Φ E **and** *in-space*: $x \in$ *uspace* Φ
shows $(x,x) \in E$
proof −
have *Id-on* (*uspace* Φ) $\subseteq E$
  using *uniformity-entourages entourage* **unfolding** *uniformity-on-def* **by** *fast*
then show *?thesis* **using** *Id-onI*[*OF in-space*] **by** *blast*
qed

lemma *smaller-entourage*:
assumes *entourage*: *entourage-in* Φ E
shows $\exists F.$ *entourage-in* Φ $F \wedge (\forall x\ y\ z.\ (x,y) \in F \wedge (y,z) \in F \longrightarrow (x,z) \in E)$
proof −
from *entourage* **obtain** F **where** *entourage-in* Φ $F \wedge F\ O\ F \subseteq E$
  using *uniformity-entourages entourage* **unfolding** *uniformity-on-def* **by** *meson*
moreover from *this* **have** $(x,z) \in E$ **if** $(x,y) \in F \wedge (y,z) \in F$ **for** $x\ y\ z$ **using** *that* **by** *blast*
ultimately show *?thesis* **by** *blast*
qed

lemma *entire-space-entourage*: *entourage-in* Φ (*uspace* $\Phi \times$ *uspace* Φ)

by (*metis entourages-exist entourage-in-space entourage-superset subset-refl*)

definition *utopology* :: *'a uniformity* ⇒ *'a topology* **where**
utopology Φ = *topology* (λ*U*. *U* ⊆ *uspace* Φ ∧ (∀ *x*∈*U*. ∃ *E*. *entourage-in* Φ *E* ∧ *E*“{*x*} ⊆ *U*))

lemma *openin-utopology* [*iff*]:
fixes Φ :: *'a uniformity*
defines *uopen U* ≡ *U* ⊆ *uspace* Φ ∧ (∀ *x*∈*U*. ∃ *E*. *entourage-in* Φ *E* ∧ *E*“{*x*} ⊆ *U*)
shows *openin* (*utopology* Φ) = *uopen*
proof −
have *uopen* (*U* ∩ *V*) **if** *hUV*: *uopen U* ∧ *uopen V* **for** *U V*
proof −
have ∃ *E*. *entourage-in* Φ *E* ∧ *E*“{*x*} ⊆ *U* ∩ *V* **if** *hx*: *x* ∈ *U* ∩ *V* **for** *x*
proof −
from *hUV hx* **obtain** E_1 E_2 **where**
entourage-in Φ E_1 ∧ *entourage-in* Φ E_2 ∧ E_1“{*x*} ⊆ *U* ∧ E_2“{*x*} ⊆ *V* **unfolding** *uopen-def* **by** *blast*
then have *entourage-in* Φ (E_1 ∩ E_2) ∧ (E_1 ∩ E_2)“{*x*} ⊆ *U* ∩ *V* **by** *blast*
then show *?thesis* **by** *fast*
qed
then show *?thesis* **using** *le-infI1 hUV* **unfolding** *uopen-def* **by** *auto*
qed
moreover have *uopen* (⋃𝒰) **if** *h𝒰*: ∀ *U*∈𝒰. *uopen U* **for** 𝒰
proof −
have ∃ *E*. *entourage-in* Φ *E* ∧ *E*“{*x*} ⊆ ⋃𝒰 **if** *hx*: *x* ∈ ⋃𝒰 **for** *x*
proof −
from *hx* **obtain** *U* **where** *hU*: *U* ∈ 𝒰 ∧ *x* ∈ *U* **by** *blast*
from *this h𝒰* **obtain** *E* **where** *entourage-in* Φ *E* ∧ *E*“{*x*} ⊆ *U* **unfolding** *uopen-def* **by** *fast*
moreover from *this hU* **have** *E*“{*x*} ⊆ ⋃𝒰 **by** *fast*
ultimately show *?thesis* **by** *blast*
qed
then show *?thesis* **using** *Union-least h𝒰* **unfolding** *uopen-def* **by** *auto*
qed
ultimately have *istopology uopen* **unfolding** *istopology-def* **by** *presburger*
from *topology-inverse'*[*OF this*] **show** *?thesis* **unfolding** *utopology-def uopen-def* **by** *blast*
qed

lemma *topspace-utopology*[*simp*]:
shows *topspace* (*utopology* Φ) = *uspace* Φ
proof −
let *?T* = *utopology* Φ
have *topspace ?T* ⊆ *uspace* Φ
using *openin-topspace openin-utopology* **by** *meson*
moreover have *openin ?T* (*uspace* Φ)
unfolding *openin-utopology* **by** (*auto intro*: *entire-space-entourage*)

ultimately show *?thesis* **using** *topspace-def* **by** *fast*
qed

definition *ucontinuous* :: $'a$ *uniformity* $\Rightarrow$ $'b$ *uniformity* $\Rightarrow$ $('a \Rightarrow 'b) \Rightarrow$ *bool*
where
ucontinuous Φ Ψ $f \longleftrightarrow$
$f \in$ *uspace* $\Phi \to$ *uspace* Ψ $\wedge$
$(\forall E.$ *entourage-in* Ψ $E \longrightarrow$ *entourage-in* Φ $\{(x, y) \in$ *uspace* $\Phi \times$ *uspace* $\Phi.$ $(f\,x,$ $f\,y) \in E\})$

lemma *ucontinuous-image-subset* [*dest*]: *ucontinuous* Φ Ψ $f \Longrightarrow f`($*uspace* $\Phi) \subseteq$ *uspace* Ψ
unfolding *ucontinuous-def* **by** *blast*

lemma *entourage-preimage-ucontinuous* [*dest*]:
assumes *ucontinuous* Φ Ψ f **and** *entourage-in* Ψ E
shows *entourage-in* Φ $\{(x, y) \in$ *uspace* $\Phi \times$ *uspace* $\Phi.$ $(f\,x, f\,y) \in E\}$
using *assms* **unfolding** *ucontinuous-def* **by** *blast*

lemma *ucontinuous-imp-continuous*:
assumes *ucontinuous* Φ Ψ f
shows *continuous-map* (*utopology* Φ) (*utopology* Ψ) f
proof (*unfold continuous-map-def*, *intro conjI allI impI*)
show $f \in$ *topspace* (*utopology* Φ) $\to$ *topspace* (*utopology* Ψ)
using *assms* **unfolding** *ucontinuous-def* **by** *auto*
next
fix U **assume** *hU*: *openin* (*utopology* Ψ) U
let $?V = \{x \in$ *topspace* (*utopology* Φ). $f\,x \in U\}$
have $\exists F.$ *entourage-in* Φ $F \wedge F``\{x\} \subseteq ?V$ **if** *hx*: $x \in$ *uspace* $\Phi \wedge f\,x \in U$ **for** x
proof −
from *that hU* **obtain** E **where** *hE*: *entourage-in* Ψ $E \wedge E``\{f\,x\} \subseteq U$
unfolding *openin-utopology* **by** *blast*
let $?F = \{(x, y) \in$ *uspace* $\Phi \times$ *uspace* $\Phi.$ $(f\,x, f\,y) \in E\}$
have $?F``\{x\} = \{y \in$ *uspace* $\Phi.$ $f\,y \in E``\{f\,x\}\}$ **unfolding** *Image-def* **using** *hx* **by** *auto*
then have $?F``\{x\} \subseteq ?V$ **using** *hE* **by** *auto*
moreover have *entourage-in* Φ $?F$
using *assms entourage-preimage-ucontinuous hE* **unfolding** *topspace-utopology* **by** *blast*
ultimately show *?thesis* **by** *blast*
qed
then show *openin* (*utopology* Φ) $?V$ **unfolding** *openin-utopology* **by** *force*
qed

1.2 Metric spaces as uniform spaces

context *Metric-space*
begin

abbreviation *mentourage* :: *real* ⇒ (*'a* × *'a*) *set* **where**
mentourage $\varepsilon \equiv \{(x,y) \in M \times M.\ d\ x\ y < \varepsilon\}$

definition *muniformity* :: *'a uniformity* **where**
muniformity = *uniformity* (M, $\lambda E.\ E \subseteq M \times M \wedge (\exists \varepsilon > 0.$ *mentourage* $\varepsilon \subseteq E))$

lemma
uspace-muniformity[*simp*]: *uspace muniformity* = M **and**
entourage-muniformity: *entourage-in muniformity* = ($\lambda E.\ E \subseteq M \times M \wedge (\exists \varepsilon > 0.$ *mentourage* $\varepsilon \subseteq E$))
proof −
have *uniformity-on* M ($\lambda E.\ E \subseteq M \times M \wedge (\exists \varepsilon > 0.$ *mentourage* $\varepsilon \subseteq E$))
unfolding *uniformity-on-def Id-on-def converse-def*
proof (*intro conjI allI impI*, *goal-cases*)
case *1*
then show *?case* **by** (*rule exI*[*of - mentourage 1*]) *force*
next
case (*5 E*)
then obtain ε **where** *hε*: $\varepsilon > 0 \wedge$ *mentourage* $\varepsilon \subseteq E$ **by** *blast*
then have $\{(y, x).\ (x, y) \in$ *mentourage* $\varepsilon\} \subseteq E$ **using** *commute* **by** *auto*
then have *mentourage* $\varepsilon \subseteq E^{-1}$ **by** *blast*
then show *?case* **using** *hε* **by** *auto*
next
case (*6 E*)
then obtain ε **where** *hε*: $\varepsilon > 0 \wedge$ *mentourage* $\varepsilon \subseteq E$ **by** *blast*
let *?F* = *mentourage* ($\varepsilon/2$)
have $(x,z) \in E$ **if** $(x,y) \in$ *?F* $\wedge$ $(y,z) \in$ *?F* **for** $x\ y\ z$
proof −
have $d\ x\ z < \varepsilon$ **using** *that triangle* **by** *fastforce*
then show *?thesis* **using** *that hε* **by** *blast*
qed
then have *?F* $\subseteq M \times M \wedge$ *?F O ?F* $\subseteq E$ **by** *blast*
then show *?case* **by** (*meson hε order-refl zero-less-divide-iff zero-less-numeral*)
next
case (*8 E F*)
then show *?case* **by** *fast*
next
case (*10 E F*)
then obtain $\varepsilon\ \delta$ **where**
$\varepsilon > 0 \wedge$ *mentourage* $\varepsilon \subseteq E$ **and**
$\delta > 0 \wedge$ *mentourage* $\delta \subseteq F$ **by** *presburger*
then have $\min\ \varepsilon\ \delta > 0 \wedge$ *mentourage* ($\min\ \varepsilon\ \delta$) $\subseteq E \cap F$ **by** *auto*
then show *?case* **by** *blast*
qed (*auto*)
then show
uspace muniformity = M **and**
entourage-in muniformity = ($\lambda E.\ E \subseteq M \times M \wedge (\exists \varepsilon > 0.$ *mentourage* $\varepsilon \subseteq E$))
unfolding *muniformity-def* **using** *uniformity-inverse'* **by** *auto*

qed

lemma *uniformity-induces-mtopology* [*simp*]: *utopology muniformity* = *mtopology*
proof −
  have *mentourage-image*: *mball x ε* = (*mentourage ε*) ‘‘ {*x*} **for** *x ε* **unfolding** *mball-def* **by** *auto*
  have *openin* (*utopology muniformity*) *U* ⟷ *openin mtopology U* **for** *U*
  proof
    assume *hU*: *openin* (*utopology muniformity*) *U*
    have ∃ *ε* > *0*. *mball x ε* ⊆ *U* **if** *x* ∈ *U* **for** *x*
    proof −
      from *hU that* **obtain** *E* **where** *hE*: *entourage-in muniformity E* ∧ *E* ‘‘ {*x*} ⊆ *U* **unfolding** *openin-utopology* **by** *blast*
      then obtain *ε* **where** *hε*: *ε* > *0* ∧ *mentourage ε* ⊆ *E* **unfolding** *entourage-muniformity* **by** *presburger*
      then have (*mentourage ε*) ‘‘ {*x*} ⊆ *U* **using** *hE* **by** *fast*
      then show *?thesis* **using** *mentourage-image hε* **by** *auto*
    qed
    then show *openin mtopology U* **unfolding** *openin-mtopology* **using** *hU openin-subset* **by** *fastforce*
  next
    assume *hU*: *openin mtopology U*
    have ∃ *E*. *entourage-in muniformity E* ∧ *E* ‘‘ {*x*} ⊆ *U* **if** *x* ∈ *U* **for** *x*
    proof −
      from *hU that* **obtain** *ε* **where** *ε* > *0* ∧ *mball x ε* ⊆ *U* **unfolding** *openin-mtopology* **by** *blast*
      then show *?thesis* **unfolding** *mentourage-image entourage-muniformity* **by** *auto*
    qed
    then show *openin* (*utopology muniformity*) *U* **unfolding** *openin-utopology* **using** *hU openin-subset* **by** *fastforce*
  qed
  then show *?thesis* **using** *topology-eq* **by** *blast*
qed

1.3 Connection to type class

end

The following connects the *uniform-space* class to the set based notion *Uniform-Structure.uniformity-on*.

Given a type *'a* which is an instance of the class *uniform-space*, it is possible to introduce an *'a uniformity* on the entire universe: *UNIV*:

definition *uniformity-of-space* :: (*'a* :: *uniform-space*) *uniformity* **where**
  uniformity-of-space = *uniformity* (*UNIV* :: *'a set*, (*λS*. ∀$_F$ *x in uniformity-class.uniformity*. *x*∈*S*))

The induced uniformity fulfills the required conditions, i.e., the class based notion implies the set-based notion.

lemma *uniformity-on-uniformity-of-space-aux*:
uniformity-on (*UNIV* :: (*'a* :: *uniform-space*) *set*) (λ*S*. ∀$_F$ *x in uniformity-class.uniformity*. *x*∈*S*)
proof −
let *?u* = *uniformity-class.uniformity* :: (*'a* × *'a*) *filter*

have ∃ *S*. (∀$_F$ *x in ?u*.*x* ∈ *S*) **by** (*intro exI*[**where** *x*=*UNIV* × *UNIV*]) *simp*
moreover have (∀$_F$ *x in ?u*.*x* ∈ *E* ∩ *F*) **if** (∀$_F$ *x in ?u*.*x* ∈ *E*) (∀$_F$ *x in ?u*.*x* ∈ *F*) **for** *E F*
using *that eventually-conj* **by** *auto*
moreover have *Id-on UNIV* ⊆ *E* **if** ∀$_F$ *x in ?u*. *x* ∈ *E* **for** *E*
proof −
have (*x*,*x*) ∈ *E* **for** *x* **using** *uniformity-refl*[*OF that*] **by** *auto*
thus *?thesis* **unfolding** *Id-on-def* **by** *auto*
qed
moreover have (∀$_F$ *x in ?u*. *x* ∈ E^{-1}) **if** ∀$_F$ *x in ?u*. *x* ∈ *E* **for** *E*
using *uniformity-sym*[*OF that*] **by** (*simp add*: *converse-unfold*)
moreover have ∃ *F*. (∀$_F$ *x in ?u*. *x* ∈ *F*) ∧ *F O F* ⊆ *E* **if** ∀$_F$ *x in ?u*. *x* ∈ *E* **for** *E*
proof −
from *uniformity-trans*[*OF that*]
obtain *D* **where** *eventually D ?u* (∀ *x y z*. *D* (*x*, *y*) ⟶ *D* (*y*, *z*) ⟶ (*x*, *z*) ∈ *E*) **by** *auto*
thus *?thesis* **by** (*intro exI*[**where** *x*=*Collect D*]) *auto*
qed
moreover have ∀$_F$ *x in ?u*. *x* ∈ *F* **if** ∀$_F$ *x in ?u*. *x* ∈ *E* *E* ⊆ *F* **for** *E F*
using *that*(*2*) **by** (*intro eventually-mono*[*OF that*(*1*)]) *auto*
ultimately show *?thesis*
unfolding *uniformity-on-def* **by** *auto*
qed

lemma *uniformity-rep-uniformity-of-space*:
uniformity-rep uniformity-of-space = (*UNIV*, (λ*S*. ∀$_F$ *x in uniformity-class.uniformity*. *x* ∈ *S*))
unfolding *uniformity-of-space-def* **using** *uniformity-on-uniformity-of-space-aux*
by (*intro uniformity-inverse*) *auto*

lemma *uspace-uniformity-space* [*simp*, *iff*]:
uspace uniformity-of-space = *UNIV*
unfolding *uspace-def uniformity-rep-uniformity-of-space* **by** *simp*

lemma *entourage-in-uniformity-space*:
entourage-in uniformity-of-space S =(∀$_F$ *x in uniformity-class.uniformity*. *x* ∈ *S*)
unfolding *entourage-in-def uniformity-rep-uniformity-of-space* **by** *simp*

Compatibility of the *Metric-space.muniformity* with the uniformity based on the class based hierarchy.

lemma (*uniformity-of-space* :: (*'a* :: *metric-space*) *uniformity*) = *Met-TC.muniformity*

proof −
  have $(\forall x\ y.\ dist\ x\ y < \varepsilon \longrightarrow (x,\ y) \in E) = (\{(x,\ y).\ dist\ x\ y < \varepsilon\} \subseteq E)$
    for ε **and** $E :: ('a \times 'a)\ set$
    by *auto*
  thus *?thesis*
   unfolding *Met-TC.muniformity-def uniformity-of-space-def eventually-uniformity-metric*
by *simp*
qed

end

2 General theory of Topological Groups

theory *Topological-Group*
  imports
    HOL−Algebra.Group
    HOL−Algebra.Coset
    HOL−Analysis.Abstract-Topology
    HOL−Analysis.Product-Topology
    HOL−Analysis.T1-Spaces
    HOL−Analysis.Abstract-Metric-Spaces
     Uniform-Structure
begin

Summary In this section we define topological groups and prove basic results about them. We also introduce the left and right uniform structures of topological groups and prove the Birkhoff-Kakutani theorem.

2.1 Auxiliary definitions and results

2.1.1 Miscellaneous

lemma *connected-components-homeo*:
  assumes *homeo*: *homeomorphic-map* T_1 T_2 φ **and** *in-space*: $x \in$ *topspace* T_1
  shows φ‘(*connected-component-of-set* T_1 x) = *connected-component-of-set* T_2 (φ x)
proof
  let *?Z* = *connected-component-of-set*
  show φ‘(*?Z* T_1 x) $\subseteq$ *?Z* T_2 (φ x)
      by (*metis connected-component-of-eq connected-component-of-maximal connectedin-connected-component-of homeo homeomorphic-map-connectedness-eq imageI in-space mem-Collect-eq*)
next
  let *?Z* = *connected-component-of-set*
  from *homeo* **obtain** ψ **where** ψ*-homeo*: *homeomorphic-map* T_2 T_1 ψ
    and ψ*-inv*: $(\forall y \in$ *topspace* $T_1.\ \psi\ (\varphi\ y) = y) \wedge (\forall y \in$ *topspace* $T_2.\ \varphi\ (\psi\ y) = y)$
    by (*smt* (*verit*) *homeomorphic-map-maps homeomorphic-maps-map*)

from *homeo in-space* **have** $\varphi\ x \in$ *topspace* T_2
using *homeomorphic-imp-surjective-map* **by** *blast*
then have ψ‘(*?Z* T_2 (φ x)) $\subseteq$ *?Z* T_1 (ψ (φ x))
by (*metis connected-component-of-eq connected-component-of-maximal connectedin-connected-component-of* ψ*-homeo homeomorphic-map-connectedness-eq imageI mem-Collect-eq*)
then show *?Z* T_2 (φ x) $\subseteq$ φ‘(*?Z* T_1 x)
by (*smt* (*verit*, *del-insts*) ψ*-inv connected-component-of-subset-topspace image-subset-iff in-space subsetD subsetI*)
qed

lemma *open-map-prod-top*:
assumes *open-map* T_1 T_3 f **and** *open-map* T_2 T_4 g
shows *open-map* (*prod-topology* T_1 T_2) (*prod-topology* T_3 T_4) ($\lambda(x, y)$. (f x, g y))
proof (*unfold open-map-def*, *standard*, *standard*)
let *?p* = $\lambda(x, y)$. (f x, g y)
fix U **assume** *openin* (*prod-topology* T_1 T_2) U
then obtain $\mathcal{U}$ **where** *h*$\mathcal{U}$: $\mathcal{U} \subseteq \{V \times W \mid V\ W.$ *openin* T_1 $V \wedge$ *openin* T_2 $W\} \wedge \bigcup \mathcal{U} = U$
unfolding *openin-prod-topology union-of-def* **using** *arbitrary-def* **by** *auto*
then have *?p*‘$U = \bigcup$ {*?p*‘$VW \mid VW.$ $VW \in \mathcal{U}$} **by** *blast*
then have *?p*‘$U = \bigcup$ {*?p*‘($V \times W$) $\mid V\ W.$ $V \times W \in \mathcal{U} \wedge$ *openin* T_1 $V \wedge$ *openin* T_2 W}
using *h*$\mathcal{U}$ **by** *blast*
moreover have *?p*‘($V \times W$) = (f‘V) $\times$ (g‘W) **for** V W **by** *fast*
ultimately have *?p*‘$U = \bigcup$ {(f‘V) $\times$ (g‘W) $\mid V\ W.$ $V \times W \in \mathcal{U} \wedge$ *openin* T_1 $V \wedge$ *openin* T_2 W} **by** *presburger*
moreover have *openin* (*prod-topology* T_3 T_4) ((f‘V) $\times$ (g‘W)) **if** *openin* T_1 V $\wedge$ *openin* T_2 W **for** V W
using *openin-prod-Times-iff assms that open-map-def* **by** *metis*
ultimately show *openin* (*prod-topology* T_3 T_4) (*?p*‘U) **by** *fastforce*
qed

lemma *injective-quotient-map-homeo*:
assumes *quotient-map T1 T2 q* **and** *inj*: *inj-on q* (*topspace T1*)
shows *homeomorphic-map T1 T2 q* **using** *assms*
unfolding *homeomorphic-eq-everything-map injective-quotient-map*[*OF inj*] **by** *fast*

lemma (**in** *group*) *subgroupI-alt*:
assumes *subset*: $H \subseteq$ *carrier* G **and** *nonempty*: $H \neq \{\}$ **and**
closed: $\bigwedge\sigma\ \tau.\ \sigma \in H \wedge \tau \in H \Longrightarrow \sigma \otimes$ *inv* $\tau \in H$
shows *subgroup* H G
proof −
from *nonempty* **obtain** η **where** $\eta \in H$ **by** *blast*
then have $\mathbf{1} \in H$ **using** *closed*[*of* η η] *subset r-inv* **by** *fastforce*
then have *closed-inv*: *inv* $\sigma \in H$ **if** $\sigma \in H$ **for** σ
using *closed*[*of* $\mathbf{1}$ σ] *r-inv r-one subset that* **by** *force*

then have $\sigma \otimes \tau \in H$ **if** $\sigma \in H \wedge \tau \in H$ **for** σ τ
using *closed*[*of* σ *inv* τ] *inv-inv subset subset-iff that* **by** *auto*
then show *?thesis* **using** *assms closed-inv* **by** (*auto intro*: *subgroupI*)
qed

lemma *subgroup-intersection*:
assumes *subgroup H G* **and** *subgroup H′ G*
shows *subgroup* ($H \cap H'$) *G*
using *assms* **unfolding** *subgroup-def* **by** *force*

2.1.2 Quotient topology

definition *quot-topology* :: *′a topology* ⇒ (*′a* ⇒ *′b*) ⇒ *′b topology* **where**
quot-topology T q = *topology* (λU. $U \subseteq q$‘(*topspace T*) ∧ *openin T* {$x \in$ *topspace T*. *q x* ∈ *U*})

lemma *quot-topology-open*:
fixes *T* :: *′a topology* **and** *q* :: *′a* ⇒ *′b*
defines *openin-quot U* ≡ $U \subseteq q$‘(*topspace T*) ∧ *openin T* {$x \in$ *topspace T*. *q x* ∈ *U*}
shows *openin* (*quot-topology T q*) = *openin-quot*
proof −
have *istopology openin-quot*
proof −
have *openin-quot* ($U_1 \cap U_2$) **if** *openin-quot* U_1 ∧ *openin-quot* U_2 **for** U_1 U_2
proof −
have {$x \in$ *topspace T*. *q x* ∈ $U_1 \cap U_2$} = {$x \in$ *topspace T*. *q x* ∈ U_1} ∩ {x ∈ *topspace T*. *q x* ∈ U_2} **by** *blast*
then show *?thesis* **using** *that* **unfolding** *openin-quot-def* **by** *auto*
qed
moreover have *openin-quot* ($\bigcup \mathcal{U}$) **if** $\forall U \in \mathcal{U}$. *openin-quot U* **for** $\mathcal{U}$
proof −
have {$x \in$ *topspace T*. *q x* ∈ $\bigcup \mathcal{U}$} = $\bigcup$ {{$x \in$ *topspace T*. *q x* ∈ *U*} | *U*. $U \in \mathcal{U}$} **by** *blast*
then show *?thesis* **using** *that* **unfolding** *openin-quot-def* **by** *auto*
qed
ultimately show *?thesis* **using** *istopology-def*
by (*smt* (*verit*) *Collect-cong Sup-set-def UnionI Union-iff image-eqI mem-Collect-eq mem-Collect-eq openin-topspace subsetI subset-antisym topspace-def*)
qed
from *topology-inverse′*[*OF this*] **show** *?thesis* **using** *quot-topology-def* **unfolding** *openin-quot-def* **by** *metis*
qed

lemma *projection-quotient-map*: *quotient-map T* (*quot-topology T q*) *q*
proof (*unfold quotient-map-def*, *intro conjI*)
have *openin* (*quot-topology T q*) (*q* ‘ *topspace T*) **using** *quot-topology-open*
by (*smt* (*verit*) *image-subset-iff mem-Collect-eq openin-subtopology-refl subsetI subtopology-superset*)

then show *q* ‘ *topspace T* = *topspace* (*quot-topology T q*) **using** *quot-topology-open*
by (*metis* (*no-types*, *opaque-lifting*) *openin-subset openin-topspace subset-antisym*)
next
show ∀ *U* ⊆ *topspace* (*quot-topology T q*).
openin T {*x* ∈ *topspace T*. *q x* ∈ *U*} = *openin* (*quot-topology T q*) *U*
using *quot-topology-open* **by** (*metis* (*mono-tags*, *lifting*) *openin-topspace order-trans*)
qed

corollary *topspace-quot-topology* [*simp*]: *topspace* (*quot-topology T q*) = *q*‘(*topspace T*)
using *projection-quotient-map quotient-imp-surjective-map* **by** *metis*

corollary *projection-continuous*: *continuous-map T* (*quot-topology T q*) *q*
using *projection-quotient-map quotient-imp-continuous-map* **by** *fast*

2.2 Definition and basic results

locale *topological-group* = *group* +
fixes *T* :: ′*g topology*
assumes *group-is-space* [*simp*]: *topspace T* = *carrier G*
assumes *inv-continuous*: *continuous-map T T* (λσ. *inv* σ)
assumes *mul-continuous*: *continuous-map* (*prod-topology T T*) *T* (λ(σ,τ). σ⊗τ)
begin

lemma *in-space-iff-in-group* [*iff*]: σ ∈ *topspace T* ⟷ σ ∈ *carrier G*
by *auto*

lemma *translations-continuous* [*intro*]:
assumes *in-group*: σ ∈ *carrier G*
shows *continuous-map T T* (λτ. σ⊗τ) **and** *continuous-map T T* (λτ. τ⊗σ)
proof −
have *continuous-map T* (*prod-topology T T*) (λτ. (σ,τ))
by (*auto intro*: *continuous-map-pairedI simp*: *in-group*)
moreover have (λτ. σ⊗τ) = (λ(σ,τ). σ⊗τ) ∘ (λτ. (σ,τ)) **by** *auto*
ultimately show *continuous-map T T* (λτ. σ⊗τ)
using *mul-continuous continuous-map-compose* **by** *metis*
next
have *continuous-map T* (*prod-topology T T*) (λτ. (τ,σ))
by (*auto intro*: *continuous-map-pairedI simp*: *in-group*)
moreover have (λτ. τ⊗σ) = (λ(σ,τ). σ⊗τ) ∘ (λτ. (τ,σ)) **by** *auto*
ultimately show *continuous-map T T* (λτ. τ⊗σ)
using *mul-continuous continuous-map-compose* **by** *metis*
qed

lemma *translations-homeos*:
assumes *in-group*: σ ∈ *carrier G*
shows *homeomorphic-map T T* (λτ. σ⊗τ) **and** *homeomorphic-map T T* (λτ. τ⊗σ)

proof −
have ∀τ∈*topspace* *T*. *inv* σ ⊗ (σ ⊗ τ) = τ **by** (*simp add*: *group.inv-solve-left′* *in-group*)
moreover have ∀τ∈*topspace* *T*. σ ⊗ (*inv* σ ⊗ τ) = τ
by (*metis group-is-space in-group inv-closed l-one m-assoc r-inv*)
ultimately have *homeomorphic-maps T T* (λτ. σ⊗τ) (λτ. (*inv* σ)⊗τ)
using *homeomorphic-maps-def in-group* **by** *blast*
then show *homeomorphic-map T T* (λτ. σ⊗τ) **using** *homeomorphic-maps-map* **by** *blast*
next
have ∀τ∈*topspace* *T*. τ ⊗ σ ⊗ *inv* σ = τ
by (*simp add*: *group.inv-solve-right′ in-group*)
moreover have ∀τ∈*topspace* *T*. τ ⊗ *inv* σ ⊗ σ = τ **by** (*simp add*: *in-group* *m-assoc*)
ultimately have *homeomorphic-maps T T* (λτ. τ⊗σ) (λτ. τ⊗(*inv* σ))
using *homeomorphic-maps-def in-group* **by** *blast*
then show *homeomorphic-map T T* (λτ. τ⊗σ) **using** *homeomorphic-maps-map* **by** *blast*
qed

abbreviation *conjugation* :: ′*g* ⇒ ′*g* ⇒ ′*g* **where**
conjugation σ τ ≡ σ ⊗ τ ⊗ *inv* σ

corollary *conjugation-homeo*:
assumes *in-group*: σ ∈ *carrier G*
shows *homeomorphic-map T T* (*conjugation* σ)
proof −
have *conjugation* σ = (λτ. τ ⊗ *inv* σ) ∘ (λτ. σ ⊗ τ) **by** *auto*
then show *?thesis* **using** *translations-homeos homeomorphic-map-compose*
by (*metis in-group inv-closed*)
qed

corollary *open-set-translations*:
assumes *open-set*: *openin T U* **and** *in-group*: σ ∈ *carrier G*
shows *openin T* (σ <# *U*) **and** *openin T* (*U* #> σ)
proof −
let *?φ* = λτ. σ ⊗ τ
have σ <# *U* = *?φ*‘*U* **unfolding** *l-coset-def* **by** *blast*
then show *openin T* (σ <# *U*) **using** *translations-homeos*[*OF in-group*]
by (*metis homeomorphic-map-openness-eq open-set*)
next
let *?ψ* = λτ. τ ⊗ σ
have *U* #> σ = *?ψ*‘*U* **unfolding** *r-coset-def* **by** *fast*
then show *openin T* (*U* #> σ) **using** *translations-homeos*[*OF in-group*]
by (*metis homeomorphic-map-openness-eq open-set*)
qed

corollary *closed-set-translations*:
assumes *closed-set*: *closedin T U* **and** *in-group*: σ ∈ *carrier G*

```
  shows closedin T (σ <# U) and closedin T (U #> σ)
proof −
  let ?φ = λτ. σ⊗τ
  have σ <# U = ?φ ‘U unfolding l-coset-def by fast
  then show closedin T (σ <# U) using translations-homeos[OF in-group]
    by (metis homeomorphic-map-closedness-eq closed-set)
next
  let ?ψ = λτ. τ⊗σ
  have U #> σ = ?ψ ‘U unfolding r-coset-def by fast
  then show closedin T (U #> σ) using translations-homeos[OF in-group]
    by (metis homeomorphic-map-closedness-eq closed-set)
qed

lemma inverse-homeo: homeomorphic-map T T (λσ. inv σ)
  using homeomorphic-map-involution[OF inv-continuous] by auto
```

2.3 Subspaces and quotient spaces

```
abbreviation connected-component-1 :: 'g set where
connected-component-1 ≡ connected-component-of-set T 1

lemma connected-component-1-props:
  shows connected-component-1 ◁ G and closedin T connected-component-1
proof −
  let ?Z = connected-component-of-set T
  have in-space: (?Z 1) ⊆ topspace T
    using connected-component-of-subset-topspace by fastforce
  have subgroup (?Z 1) G
  proof (rule subgroupI)
    show (?Z 1) ⊆ carrier G using in-space by auto
  next
    show (?Z 1) ≠ {}
      by (metis connected-component-of-eq-empty group-is-space one-closed)
  next
    fix σ assume hσ: σ ∈ (?Z 1)
    let ?φ = λη. inv η
    have ?φ‘(?Z 1) = ?Z 1 using connected-components-homeo
      by (metis group-is-space inv-one inverse-homeo one-closed)
    then show inv σ ∈ (?Z 1) using hσ by blast
  next
    fix σ τ assume hσ: σ ∈ (?Z 1) and hτ: τ ∈ (?Z 1)
    let ?φ = λη. σ ⊗ η
    have ?φ‘(?Z 1) = ?Z σ using connected-components-homeo
        by (metis group-is-space hσ in-space one-closed r-one subset-eq transla-
tions-homeos(1))
    moreover have ?Z σ = ?Z 1 using hσ by (simp add: connected-component-of-equiv)
    ultimately show σ ⊗ τ ∈ ?Z 1 using hτ by blast
  qed
  moreover have conjugation σ τ ∈ ?Z 1 if hστ: σ ∈ carrier G ∧ τ ∈ ?Z 1 for
```

σ τ
  proof –
    let *?φ* = *conjugation σ*
    have *?φ*‘(*?Z* **1**) = *?Z* (*?φ* **1**) **using** *connected-components-homeo*
      by (*metis conjugation-homeo group-is-space one-closed hστ*)
    then show *?thesis* **using** *r-inv r-one hστ* **by** *auto*
  qed
  ultimately show *connected-component-1* ◁ *G* **using** *normal-inv-iff* **by** *blast*
next
  show *closedin T connected-component-1* **by** (*simp add*: *closedin-connected-component-of*)
qed

lemma *group-prod-space* [*simp*]: *topspace* (*prod-topology T T*) = (*carrier G*) × (*carrier G*)
  by *auto*

no-notation *eq-closure-of* (‹*closure′-of*1›)

lemma *subgroup-closure*:
  assumes *H-subgroup*: *subgroup H G*
  shows *subgroup* (*T closure-of H*) *G*
proof –
  have *subset*: *T closure-of H* ⊆ *carrier G*
    by (*metis closedin-closure-of closedin-subset group-is-space*)
  have *nonempty*: *T closure-of H* ≠ {}
    by (*simp add*: *assms closure-of-eq-empty group.subgroupE*(*1*) *subgroupE*(*2*))

  let *?φ* = λ(σ,τ). σ ⊗ *inv* τ
  have *φ-continuous*: *continuous-map* (*prod-topology T T*) *T ?φ*
  proof –
    have *continuous-map* (*prod-topology T T*) (*prod-topology T T*) (λ(σ, τ). (σ, *inv* τ))
      using *continuous-map-prod-top inv-continuous* **by** *fastforce*
    moreover have *?φ* = (λ(σ, τ). σ ⊗ τ) ∘ (λ(σ, τ). (σ, *inv* τ)) **by** *fastforce*
    ultimately show *?thesis* **using** *mul-continuous continuous-map-compose* **by** *force*
  qed

  have σ ⊗ *inv* τ ∈ *T closure-of H*
    if *hστ*: σ ∈ *T closure-of H* ∧ τ ∈ *T closure-of H* **for** σ τ
  proof –
    have *in-space*: σ ⊗ *inv* τ ∈ *topspace T* **using** *subset hστ* **by** *fast*
    have ∃η ∈ *H*. η ∈ *U* **if** *hU*: *openin T U* ∧ σ ⊗ *inv* τ ∈ *U* **for** *U*
    proof –
      let *?V* = {*x* ∈ *topspace* (*prod-topology T T*). *?φ x* ∈ *U*}
      have *openin* (*prod-topology T T*) *?V*
        using *φ-continuous hU openin-continuous-map-preimage* **by** *blast*
      moreover have (σ,τ) ∈ *?V*
        using *hU group-prod-space hστ subset* **by** *force*

ultimately obtain V_1 V_2 **where**
hV_1V_2: *openin* T V_1 $\wedge$ *openin* T V_2 $\wedge$ $\sigma \in V_1 \wedge \tau \in V_2 \wedge V_1 \times V_2 \subseteq$ *?V*
by (*smt* (*verit*) *openin-prod-topology-alt*)
then obtain σ' τ' **where** $h\sigma'\tau'$: $\sigma' \in V_1 \cap H \wedge \tau' \in V_2 \cap H$ **using** $h\sigma\tau$
by (*meson all-not-in-conv disjoint-iff openin-Int-closure-of-eq-empty*)
then have *?φ* $(\sigma',\tau') \in U$ **using** hV_1V_2 **by** *blast*
moreover have *?φ* $(\sigma',\tau') \in H$ **using** $h\sigma'\tau'$ *H-subgroup subgroupE*(*3*,*4*) **by** *simp*
ultimately show *?thesis* **by** *blast*
qed
then show *?thesis* **using** *closure-of-def in-space* **by** *force*
qed
then show *?thesis* **using** *subgroupI-alt subset nonempty* **by** *blast*
qed

lemma *normal-subgroup-closure*:
assumes *normal-subgroup*: $N \lhd G$
shows (T *closure-of* N) $\lhd$ G
proof −
have (*conjugation* σ)‘(T *closure-of* N) $\subseteq$ T *closure-of* N **if** $h\sigma$: $\sigma \in$ *carrier* G **for** σ
proof −
have (*conjugation* σ)‘$N \subseteq N$ **using** *normal-subgroup normal-invE*(*2*) $h\sigma$ **by** *auto*
then have T *closure-of* (*conjugation* σ)‘$N \subseteq T$ *closure-of* N
using *closure-of-mono* **by** *meson*
moreover have (*conjugation* σ)‘(T *closure-of* N) $\subseteq$ T *closure-of* (*conjugation* σ)‘N
using $h\sigma$ *conjugation-homeo*
by (*meson continuous-map-eq-image-closure-subset homeomorphic-imp-continuous-map*)
ultimately show *?thesis* **by** *blast*
qed
moreover have *subgroup* (T *closure-of* N) G **using** *subgroup-closure*
by (*simp add*: *normal-invE*(*1*) *normal-subgroup*)
ultimately show *?thesis* **using** *normal-inv-iff* **by** *auto*
qed

lemma *topological-subgroup*:
assumes *subgroup* H G
shows *topological-group* (G (|*carrier* := H|)) (*subtopology* T H)
proof −
interpret *subgroup* H G **by** *fact*
let *?ℋ* = (G (|*carrier* := H|)) **and** *?T'* = *subtopology* T H
have *H-subspace*: *topspace ?T'* = H **using** *topspace-subtopology-subset* **by** *force*
have *continuous-map ?T'* T (λσ. *inv* σ) **using** *continuous-map-from-subtopology inv-continuous* **by** *blast*
moreover have (λσ. *inv* σ) $\in$ *topspace ?T'* $\to$ H **unfolding** *Pi-def H-subspace* **by** *blast*
ultimately have *continuous-map ?T' ?T'* (λσ. *inv* σ) **using** *continuous-map-into-subtopology*

by *blast*
then have *sub-inv-continuous*: *continuous-map ?T′ ?T′* ($\lambda\sigma$. $inv_{?\mathcal{H}}$ σ)
using *continuous-map-eq H-subspace m-inv-consistent assms* **by** *fastforce*
have *continuous-map* (*prod-topology ?T′ ?T′*) *T* ($\lambda(\sigma,\tau)$. $\sigma \otimes \tau$)
unfolding *subtopology-Times*[*symmetric*] **using** *continuous-map-from-subtopology*[*OF mul-continuous*] **by** *fast*
moreover have ($\lambda(\sigma,\tau)$. $\sigma \otimes \tau$) ∈ *topspace* (*prod-topology ?T′ ?T′*) → *H*
unfolding *Pi-def topspace-prod-topology H-subspace* **by** *fast*
ultimately have *continuous-map* (*prod-topology ?T′ ?T′*) *?T′* ($\lambda(\sigma,\tau)$. $\sigma \otimes \tau$)
using *continuous-map-into-subtopology* **by** *blast*
then have *continuous-map* (*prod-topology ?T′ ?T′*) *?T′* ($\lambda(\sigma,\tau)$. $\sigma \otimes_{?\mathcal{H}} \tau$) **by** *fastforce*
then show *?thesis* **unfolding** *topological-group-def topological-group-axioms-def* **using** *H-subspace sub-inv-continuous* **by** *auto*
qed

Topology on the set of cosets of some subgroup

abbreviation *coset-topology* :: *′g set* ⇒ *′g set topology* **where**
coset-topology H ≡ *quot-topology T* (*r-coset G H*)

lemma *coset-topology-topspace*[*simp*]:
shows *topspace* (*coset-topology H*) = (*r-coset G H*)‘(*carrier G*)
using *projection-quotient-map quotient-imp-surjective-map group-is-space* **by** *metis*

lemma *projection-open-map*:
assumes *subgroup*: *subgroup H G*
shows *open-map T* (*coset-topology H*) (*r-coset G H*)
proof (*unfold open-map-def*, *standard*, *standard*)
fix *U* **assume** *hU*: *openin T U*
let *?π* = *r-coset G H*
let *?V* = {σ ∈ *topspace T*. *?π* σ ∈ *?π*‘*U*}
have *subsets*: *H* ⊆ *carrier G* ∧ *U* ⊆ *carrier G*
using *subgroup hU openin-subset* **by** (*force elim*!: *subgroupE*)
have *?V* = {σ ∈ *carrier G*. ∃τ ∈ *U*. *H* #> σ = *H* #> τ} **using** *image-def* **by** *blast*
then have *?V* = {σ ∈ *carrier G*. ∃τ ∈ *U*. σ ∈ *H* #> τ} **using** *subsets*
by (*smt* (*verit*) *Collect-cong rcos-self repr-independence subgroup subset-eq*)
also have ... = (⋃η ∈ *H*. η <# *U*) **unfolding** *r-coset-def l-coset-def* **using** *subsets* **by** *auto*
moreover have *openin T* (η <# *U*) **if** η ∈ *H* **for** η
using *open-set-translations*(*1*)[*OF hU*] *subsets that* **by** *blast*
ultimately have *openin T ?V* **by** *fastforce*
then show *openin* (*coset-topology H*) (*?π*‘*U*) **using** *quot-topology-open hU*
by (*metis* (*mono-tags*, *lifting*) *Collect-cong image-mono openin-subset*)
qed

lemma *topological-quotient-group*:
assumes *normal-subgroup*: *N* ◁ *G*
shows *topological-group* (*G Mod N*) (*coset-topology N*)

```
proof −
  interpret normal N G by fact
  let ?π = r-coset G N
  let ?T′ = coset-topology N
  have quot-space: topspace ?T′ = ?π‘(carrier G) using coset-topology-topspace by presburger
  then have quot-group-quot-space: topspace ?T′ = carrier (G Mod N) using carrier-FactGroup by metis

  let ?quot-mul = λ(Nσ, Nτ). Nσ ⊗G Mod N Nτ
  have π-prod-space: topspace (prod-topology ?T′ ?T′) = ?π‘(carrier G) × ?π‘(carrier G)
    using quot-space topspace-prod-topology by simp
  have quot-mul-continuous: continuous-map (prod-topology ?T′ ?T′) ?T′ ?quot-mul
  proof (unfold continuous-map-def, intro conjI ballI allI impI)
    show ?quot-mul ∈ topspace (prod-topology ?T′ ?T′) → topspace ?T′
      using rcos-sum unfolding quot-space π-prod-space by auto
  next
    fix U assume hU: openin ?T′ U
    let ?V = {p ∈ topspace (prod-topology ?T′ ?T′). ?quot-mul p ∈ U}
    let ?W = {(σ,τ) ∈ topspace (prod-topology T T). N #> (σ ⊗ τ) ∈ U}
    let ?π2 = λ(σ, τ). (N #> σ, N #> τ)
    have (λ(σ,τ). N #> (σ ⊗ τ)) = ?π ∘ (λ(σ,τ). σ ⊗ τ) by fastforce
    then have continuous-map (prod-topology T T) ?T′ (λ(σ,τ). N #> (σ ⊗ τ))
      using continuous-map-compose mul-continuous projection-continuous by fastforce
    then have openin (prod-topology T T) ?W
      using hU openin-continuous-map-preimage
      by (smt (verit) Collect-cong case-prodE case-prodI2 case-prod-conv)
    moreover have open-map (prod-topology T T) (prod-topology ?T′ ?T′) ?π2
      using projection-open-map open-map-prod-top by (metis subgroup-axioms)
    ultimately have openin (prod-topology ?T′ ?T′) (?π2‘?W) using open-map-def by blast
    moreover have ?V = ?π2‘?W
      using rcos-sum unfolding π-prod-space group-prod-space by auto
    ultimately show openin (prod-topology ?T′ ?T′) ?V by presburger
  qed

  let ?quot-inv = λNσ. invG Mod N Nσ
  have π-inv: ?quot-inv (N #> σ) = ?π (inv σ) if σ ∈ carrier G for σ
    using inv-FactGroup rcos-inv carrier-FactGroup that by blast
  have continuous-map ?T′ ?T′ ?quot-inv
  proof (unfold continuous-map-def, intro conjI ballI allI impI)
    show ?quot-inv ∈ topspace ?T′ → topspace ?T′ using π-inv quot-space by auto
  next
    fix U assume hU: openin ?T′ U
    let ?V = {Nσ ∈ topspace ?T′. ?quot-inv Nσ ∈ U}
    let ?W = {σ ∈ topspace T. N #> (inv σ) ∈ U}
    have (λσ. N #> (inv σ)) = ?π ∘ (λσ. inv σ) by fastforce
```

then have *continuous-map T ?T′* (λσ. *N* #> (*inv* σ))
using *continuous-map-compose projection-continuous inv-continuous*
by (*metis* (*no-types*, *lifting*))
then have *openin T ?W* **using** *hU openin-continuous-map-preimage* **by** *blast*
then have *openin ?T′* (*?π*‘*?W*)
using *projection-open-map* **by** (*simp add*: *open-map-def subgroup-axioms*)
moreover have *?V* = *?π*‘*?W* **using** *π-inv quot-space* **by** *force*
ultimately show *openin ?T′ ?V* **by** *presburger*
qed

then show *?thesis* **unfolding** *topological-group-def topological-group-axioms-def*
using *quot-group-quot-space quot-mul-continuous factorgroup-is-group* **by** *blast*
qed

See [3] for our approach to proving that quotient groups of topological groups are topological.

abbreviation *neighborhood* :: ′*g* ⇒ ′*g set* ⇒ *bool* **where**
neighborhood σ *U* ≡ *openin T U* ∧ σ ∈ *U*

abbreviation *symmetric* :: ′*g set* ⇒ *bool* **where**
symmetric S ≡ {*inv* σ | σ. σ ∈ *S*} ⊆ *S*

Note that this implies the other inclusion, so symmetric subsets are equal to their image under inversion.

lemma *neighborhoods-of-1*:
assumes *neighborhood* **1** *U*
shows ∃ *V*. *neighborhood* **1** *V* ∧ *symmetric V* ∧ *V* <#> *V* ⊆ *U*
proof −
have *a*: ∃ *V*⊆*U*′. *neighborhood* **1** *V* ∧ *symmetric V* **if** *hU*′: *neighborhood* **1** *U*′
for *U*′
proof −
let *?W* = {σ ∈ *carrier G*. *inv* σ ∈ *U*′}
let *?V* = *?W* ∩ ((λσ. *inv* σ)‘*?W*)
have *neighborhood* **1** *?W* **using** *openin-continuous-map-preimage*[*OF inv-continuous*]
hU′ *inv-one* **by** *fastforce*
moreover from *this* **have** *neighborhood* **1** ((λσ. *inv* σ)‘*?W*) **using** *inverse-homeo*

homeomorphic-imp-open-map inv-one image-eqI open-map-def **by** (*metis*
(*mono-tags*, *lifting*))
ultimately have *neighborhood*: *neighborhood* **1** *?V* **by** *blast*
have *inv* σ ∈ *?V* **if** σ ∈ *?V* **for** σ **using** *that* **by** *auto*
then have *symmetric ?V* **by** *fast*
moreover have σ ∈ *U*′ **if** σ ∈ *?V* **for** σ **using** *that* **by** *blast*
ultimately show *?thesis* **using** *neighborhood* **by** *blast*
qed
have *b*: ∃ *V*. *neighborhood* **1** *V* ∧ *V* <#> *V* ⊆ *U*′ **if** *hU*′: *neighborhood* **1** *U*′
for *U*′
proof −
let *?W* = {(σ,τ) ∈ *carrier G* × *carrier G*. σ⊗τ ∈ *U*′}

have *preimage-mul*: *?W* = {*x* ∈ *topspace* (*prod-topology T T*). (λ(σ,τ). σ⊗τ) *x* ∈ *U'*}
using *topspace-prod-topology* **by** *fastforce*
then have *openin* (*prod-topology T T*) *?W* ∧ (**1**,**1**) ∈ *?W*
using *openin-continuous-map-preimage*[*OF mul-continuous*] *hU' r-one* **by** *fastforce*
then obtain W_1 W_2 **where** hW_1W_2: *neighborhood* **1** W_1 ∧ *neighborhood* **1** W_2 ∧ $W_1 \times W_2 \subseteq$ *?W*
using *openin-prod-topology-alt*[**where** *S*=*?W*] **by** *meson*
let *?V* = $W_1 \cap W_2$
from hW_1W_2 **have** *neighborhood* **1** *?V* **by** *fast*
moreover have σ⊗τ ∈ *U'* **if** σ∈*?V* ∧ τ∈*?V* **for** σ τ **using** *preimage-mul* hW_1W_2 *that* **by** *blast*
ultimately show *?thesis* **unfolding** *set-mult-def* **by** *blast*
qed
from *b*[*OF assms*] **obtain** *W* **where** *hW*: *neighborhood* **1** *W* ∧ *W* <#> *W* ⊆ *U* **by** *presburger*
from *this a* **obtain** *V* **where** *V*⊆*W* ∧ *neighborhood* **1** *V* ∧ *symmetric V* **by** *presburger*
moreover from *this* **have** *V* <#> *V* ⊆ *U* **using** *hW mono-set-mult* **by** *blast*
ultimately show *?thesis* **unfolding** *set-mult-def* **by** *blast*
qed

lemma *Hausdorff-coset-space*:
assumes *subgroup*: *subgroup H G* **and** *H-closed*: *closedin T H*
shows *Hausdorff-space* (*coset-topology H*)
proof (*unfold Hausdorff-space-def*, *intro allI impI*)
interpret *subgroup H G* **by** *fact*
let *?π* = *r-coset G H*
let *?T'* = *coset-topology H*
fix *Hσ Hτ* **assume** *cosets*: *Hσ* ∈ *topspace ?T'* ∧ *Hτ* ∈ *topspace ?T'* ∧ *Hσ* ≠ *Hτ*
then obtain σ τ **where** *hστ*: σ ∈ *carrier G* ∧ τ ∈ *carrier G* ∧ *Hσ* = *H* #> σ ∧ *Hτ* = *H* #> τ **by** *auto*
then have σ ∉ *H* #> τ **using** *cosets subgroup repr-independence* **by** *blast*
have **1** ∉ (*inv* σ) <# (*H* #> τ)
proof
assume **1** ∈ *inv* σ <# (*H* #> τ)
then obtain η **where** *hη*: η ∈ *H* ∧ **1** = (*inv* σ) ⊗ (η ⊗ τ) **unfolding** *r-coset-def l-coset-def* **by** *auto*
then have σ = η ⊗ τ
by (*metis* (*no-types*, *lifting*) *Units-eq Units-m-closed group.inv-comm group-l-invI hστ inv-closed inv-inv inv-unique' l-inv-ex mem-carrier*)
then show *False* **using** ‹σ ∉ *H* #> τ› *hη r-coset-def* **by** *fast*
qed
let *?U* = *topspace T* − ((*inv* σ) <# (*H* #> τ))
have *closedin T* ((*inv* σ) <# (*H* #> τ))
using *closed-set-translations closed-set-translations*[*OF H-closed*] *hστ* **by** *simp*
then have *neighborhood* **1** *?U* **using** ‹**1** ∉ (*inv* σ) <# (*H* #> τ)› **by** *blast*

```
  then obtain V where hV: neighborhood 1 V ∧ symmetric V ∧ V <#> V ⊆
?U
    using neighborhoods-of-1 by presburger
  let ?V1 = σ <# V and ?V2 = τ <# V
  have disjoint: ?π‘?V1 ∩ ?π‘?V2 = {}
  proof (rule ccontr)
    assume ?π‘?V1 ∩ ?π‘?V2 ≠ {}
    then obtain υ1 υ2 where hυ1υ2: υ1 ∈ V ∧ υ2 ∈ V ∧ ?π (σ⊗υ1) = ?π (τ⊗υ2)

      unfolding l-coset-def by auto
    moreover then have υ1υ2-in-group: υ1 ∈ carrier G ∧ υ2 ∈ carrier G
      using hV openin-subset by force
    ultimately have in-H: (σ⊗υ1) ⊗ inv (τ⊗υ2) ∈ H
      using subgroup repr-independenceD rcos-module-imp hστ m-closed
      by (metis group.rcos-self is-group subgroup.m-closed subgroup-self)
    let ?η = (σ⊗υ1) ⊗ inv (τ⊗υ2)
    have ?η = σ ⊗ (υ1 ⊗ inv υ2) ⊗ inv τ using hστ υ1υ2-in-group m-assoc
      by (simp add: inv-mult-group subgroupE(4) subgroup-self)
    then have inv σ ⊗ (?η ⊗ τ) = υ1 ⊗ inv υ2
      using hστ υ1υ2-in-group m-assoc inv-solve-left' by auto
    then have υ1 ⊗ inv υ2 ∈ (inv σ) <# (H #> τ)
      unfolding l-coset-def r-coset-def using hστ inv-closed in-H by force
    moreover have υ1 ⊗ inv υ2 ∈ ?U using hυ1υ2 hV unfolding set-mult-def
by blast
    ultimately show False by force
  qed
  have neighborhood σ ?V1 ∧ neighborhood τ ?V2
    using open-set-translations[of V] l-coset-def hV hστ r-one by force
  then have openin ?T' (?π‘?V1) ∧ openin ?T' (?π‘?V2) ∧ Hσ ∈ ?π‘?V1 ∧ Hτ
∈ ?π‘?V2
    using projection-open-map open-map-def subgroup hστ by fast
  then show ∃ W1 W2. openin ?T' W1 ∧ openin ?T' W2 ∧ Hσ ∈ W1 ∧ Hτ ∈
W2 ∧ disjnt W1 W2
    using disjoint disjnt-def by meson
qed

lemma Hausdorff-coset-space-converse:
  assumes subgroup: subgroup H G
  assumes Hausdorff: Hausdorff-space (coset-topology H)
  shows closedin T H
proof −
  interpret subgroup H G by fact
  let ?T' = coset-topology H
  have H ∈ topspace ?T' using coset-topology-topspace coset-join2[of 1 H] sub-
group by auto
  then have closedin ?T' {H}
   using t1-space-closedin-singleton Hausdorff-imp-t1-space[OF Hausdorff] by fast
  then have preimage-closed: closedin T {σ ∈ carrier G. H #> σ = H}
    using projection-continuous closedin-continuous-map-preimage by fastforce
```

have *σ* ∈ *H* ⟷ *H* #> *σ* = *H* **if** *σ* ∈ *carrier G* **for** *σ*
using *coset-join1 coset-join2 subgroup that* **by** *metis*
then have *H* = {*σ* ∈ *carrier G*. *H* #> *σ* = *H*} **using** *subset* **by** *auto*
then show *?thesis* **using** *preimage-closed* **by** *presburger*
qed

corollary *Hausdorff-coset-space-iff*:
assumes *subgroup*: *subgroup H G*
shows *Hausdorff-space* (*coset-topology H*) ⟷ *closedin T H*
using *Hausdorff-coset-space Hausdorff-coset-space-converse subgroup* **by** *blast*

corollary *topological-group-hausdorff-iff-one-closed*:
shows *Hausdorff-space T* ⟷ *closedin T* {**1**}
proof −
let *?π* = *r-coset G* {**1**}
have *inj-on ?π* (*carrier G*) **unfolding** *inj-on-def r-coset-def* **by** *simp*
then have *homeomorphic-map T* (*coset-topology* {**1**}) *?π*
using *projection-quotient-map injective-quotient-map-homeo group-is-space* **by** *metis*
then have *Hausdorff-space T* ⟷ *Hausdorff-space* (*coset-topology* {**1**})
using *homeomorphic-Hausdorff-space homeomorphic-map-imp-homeomorphic-space* **by** *blast*
then show *?thesis* **using** *Hausdorff-coset-space-iff triv-subgroup* **by** *blast*
qed

lemma *set-mult-one-subset*:
assumes *A* ⊆ *carrier G* ∧ *B* ⊆ *carrier G* **and** **1** ∈ *B*
shows *A* ⊆ *A* <#> *B*
unfolding *set-mult-def* **using** *assms r-one* **by** *force*

lemma *open-set-mult-open*:
assumes *openin T U* ∧ *S* ⊆ *carrier G*
shows *openin T* (*S* <#> *U*)
proof −
have *S* <#> *U* = (⋃*σ*∈*S*. *σ* <# *U*) **unfolding** *set-mult-def l-coset-def* **by** *blast*
moreover have *openin T* (*σ* <# *U*) **if** *σ* ∈ *S* **for** *σ* **using** *open-set-translations*(*1*) *assms that* **by** *auto*
ultimately show *?thesis* **by** *auto*
qed

lemma *open-set-inv-open*:
assumes *openin T U*
shows *openin T* (*set-inv U*)
proof −
have *set-inv U* = (*λσ*. *inv σ*)‘*U* **unfolding** *image-def SET-INV-def* **by** *blast*
then show *?thesis* **using** *inverse-homeo homeomorphic-imp-open-map open-map-def assms* **by** *metis*
qed

lemma *open-set-in-carrier*[*elim*]:
assumes *openin T U*
shows $U \subseteq$ *carrier G*
using *openin-subset assms* **by** *force*

2.4 Uniform structures

abbreviation *left-entourage* :: *'g set* $\Rightarrow$ (*'g* $\times$ *'g*) *set* **where**
left-entourage U $\equiv$ $\{(\sigma,\tau) \in$ *carrier G* $\times$ *carrier G*. *inv* $\sigma \otimes \tau \in U\}$

abbreviation *right-entourage* :: *'g set* $\Rightarrow$ (*'g* $\times$ *'g*) *set* **where**
right-entourage U $\equiv$ $\{(\sigma,\tau) \in$ *carrier G* $\times$ *carrier G*. $\sigma \otimes$ *inv* $\tau \in U\}$

definition *left-uniformity* :: *'g uniformity* **where** *left-uniformity* =
uniformity (*carrier G*, λE. $E \subseteq$ *carrier G* $\times$ *carrier G* $\wedge$ ($\exists U$. *neighborhood* **1** $U \wedge$ *left-entourage U* $\subseteq E$))

definition *right-uniformity* :: *'g uniformity* **where** *right-uniformity* =
uniformity (*carrier G*, λE. $E \subseteq$ *carrier G* $\times$ *carrier G* $\wedge$ ($\exists U$. *neighborhood* **1** $U \wedge$ *right-entourage U* $\subseteq E$))

lemma
uspace-left-uniformity[*simp*]: *uspace left-uniformity* = *carrier G* (**is** *?space-def*)
and
entourage-left-uniformity: *entourage-in left-uniformity* =
(λE. $E \subseteq$ *carrier G* $\times$ *carrier G* $\wedge$ ($\exists U$. *neighborhood* **1** $U \wedge$ *left-entourage U* $\subseteq E$)) (**is** *?entourage-def*)
proof −
let *?*Φ = λE. $E \subseteq$ *carrier G* $\times$ *carrier G* $\wedge$ ($\exists U$. *neighborhood* **1** $U \wedge$ *left-entourage* $U \subseteq E$)
have *?*Φ (*carrier G* $\times$ *carrier G*)
using *exI*[**where** *x*=*carrier G*] *openin-topspace* **by** *force*
moreover have *Id-on* (*carrier G*) $\subseteq E \wedge$ *?*Φ (E^{-1}) $\wedge$ ($\exists F$. *?*Φ $F \wedge F$ *O* $F \subseteq E$) $\wedge$
($\forall F$. $E \subseteq F \wedge F \subseteq$ *carrier G* $\times$ *carrier G* $\longrightarrow$ *?*Φ F) **if** *hE*: *?*Φ E **for** E
proof −
from *hE* **obtain** U **where** *hU*: *neighborhood* **1** $U \wedge$ *left-entourage* $U \subseteq E$ **by** *presburger*
then have *U-subset*: $U \subseteq$ *carrier G* **using** *openin-subset* **by** *force*
from *hU* **have** *Id-on* (*carrier G*) $\subseteq E$ **by** *fastforce*
moreover have *?*Φ (E^{-1})
proof −
have $(\tau,\sigma) \in E$ **if** $\sigma \in$ *carrier G* $\wedge$ $\tau \in$ *carrier G* $\wedge$ *inv* $\sigma \otimes \tau \in$ *set-inv U* **for** σ τ
proof −
have *inv* $\tau \otimes \sigma$ = *inv* (*inv* $\sigma \otimes \tau$) **using** *that inv-mult-group* **by** *auto*
from *this* **have** *inv* $\tau \otimes \sigma \in U$ **using** *that inv-inv U-subset* **unfolding** *SET-INV-def* **by** *auto*

then show *?thesis* **using** *that hU* **by** *fast*
qed
then have *left-entourage* (*set-inv U*) ⊆ E^{-1} **by** *blast*
moreover have *neighborhood* **1** (*set-inv U*) **using** *inv-one hU open-set-inv-open SET-INV-def* **by** *fastforce*
ultimately show *?thesis* **using** *hE* **by** *auto*
qed
moreover have ∃ *F*. *?Φ F* ∧ *F O F* ⊆ *E*
proof −
obtain *V* **where** *hV*: *neighborhood* **1** *V* ∧ *V* <#> *V* ⊆ *U*
using *neighborhoods-of-1 hU* **by** *meson*
let *?F* = *left-entourage V*
have (σ,ϱ) ∈ *E* **if** (σ,τ) ∈ *?F* ∧ (τ,ϱ) ∈ *?F* **for** σ τ ϱ
proof −
have σ ∈ *carrier G* ∧ τ ∈ *carrier G* ∧ ϱ ∈ *carrier G* **using** *that* **by** *force*
then have *inv* σ ⊗ ϱ = (*inv* σ ⊗ τ) ⊗ (*inv* τ ⊗ ϱ)
using *m-assoc inv-closed m-closed r-inv r-one* **by** *metis*
moreover have (*inv* σ ⊗ τ) ⊗ (*inv* τ ⊗ ϱ) ∈ *U* **using** *that hV* **unfolding** *set-mult-def* **by** *fast*
ultimately show *?thesis* **using** *hU that* **by** *force*
qed
moreover have *?Φ ?F* **using** *hV* **by** *blast*
ultimately show *?thesis* **using** *hV* **by** *auto*
qed
moreover have ∀ *F*. *E* ⊆ *F* ∧ *F* ⊆ *carrier G* × *carrier G* ⟶ *?Φ F* **using** *hE* **by** *auto*
ultimately show *?thesis* **by** *blast*
qed
moreover have *?Φ* (*E* ∩ *F*) **if** *hEF*: *?Φ E* ∧ *?Φ F* **for** *E F*
proof −
from *hEF* **obtain** *U V* **where**
hU: *neighborhood* **1** *U* ∧ *left-entourage U* ⊆ *E* **and**
hV: *neighborhood* **1** *V* ∧ *left-entourage V* ⊆ *F* **by** *presburger*
then have *neighborhood* **1** (*U* ∩ *V*) ∧ *left-entourage* (*U* ∩ *V*) ⊆ *E* ∩ *F* **by** *fast*
then show *?thesis* **using** *that* **by** *auto*
qed
ultimately have *uniformity-on* (*carrier G*) *?Φ*
unfolding *uniformity-on-def* **by** *auto*
from *uniformity-inverse*′[*OF this*] **show** *?space-def* **and** *?entourage-def* **unfolding** *left-uniformity-def* **by** *auto*
qed

lemma
uspace-right-uniformity[*simp*]: *uspace right-uniformity* = *carrier G* (**is** *?space-def*) **and**
entourage-right-uniformity: *entourage-in right-uniformity* =
(λ*E*. *E* ⊆ *carrier G* × *carrier G* ∧ (∃ *U*. *neighborhood* **1** *U* ∧ *right-entourage U* ⊆ *E*)) (**is** *?entourage-def*)

proof −
let *?Φ* = λ*E*. *E* ⊆ *carrier G* × *carrier G* ∧ (∃ *U*. *neighborhood* **1** *U* ∧ *right-entourage U* ⊆ *E*)
have *?Φ* (*carrier G* × *carrier G*)
using *exI*[**where** *x*=*carrier G*] *openin-topspace* **by** *force*
moreover have *Id-on* (*carrier G*) ⊆ *E* ∧ *?Φ* (E^{-1}) ∧ (∃ *F*. *?Φ F* ∧ *F O F* ⊆ *E*) ∧
(∀ *F*. *E* ⊆ *F* ∧ *F* ⊆ *carrier G* × *carrier G* ⟶ *?Φ F*) **if** *hE*: *?Φ E* **for** *E*
proof −
from *hE* **obtain** *U* **where**
hU: *neighborhood* **1** *U* ∧ *right-entourage U* ⊆ *E*
by *presburger*
then have *U-subset*: *U* ⊆ *carrier G* **using** *openin-subset* **by** *force*
from *hU* **have** *Id-on* (*carrier G*) ⊆ *E* **by** *fastforce*
moreover have *?Φ* (E^{-1})
proof −
have (τ,σ) ∈ *E* **if** σ ∈ *carrier G* ∧ τ ∈ *carrier G* ∧ σ ⊗ *inv* τ ∈ *set-inv U* **for** σ τ
proof −
have τ ⊗ *inv* σ = *inv* (σ ⊗ *inv* τ) **using** *that inv-mult-group* **by** *auto*
from *this* **have** τ ⊗ *inv* σ ∈ *U* **using** *that inv-inv U-subset* **unfolding** *SET-INV-def* **by** *auto*
then show *?thesis* **using** *that hU* **by** *fast*
qed
then have *right-entourage* (*set-inv U*) ⊆ E^{-1} **by** *blast*
moreover have *neighborhood* **1** (*set-inv U*) **using** *inv-one hU open-set-inv-open SET-INV-def* **by** *fastforce*
ultimately show *?thesis* **using** *hE* **by** *auto*
qed
moreover have ∃ *F*. *?Φ F* ∧ *F O F* ⊆ *E*
proof −
obtain *V* **where** *hV*: *neighborhood* **1** *V* ∧ *V* <#> *V* ⊆ *U*
using *neighborhoods-of-1 hU* **by** *meson*
let *?F* = *right-entourage V*
have (σ,ϱ) ∈ *E* **if** (σ,τ) ∈ *?F* ∧ (τ,ϱ) ∈ *?F* **for** σ τ ϱ
proof −
have σ ∈ *carrier G* ∧ τ ∈ *carrier G* ∧ ϱ ∈ *carrier G* **using** *that* **by** *force*
then have σ ⊗ *inv* ϱ = (σ ⊗ *inv* τ) ⊗ (τ ⊗ *inv* ϱ)
using *m-assoc inv-closed m-closed l-inv r-one* **by** *metis*
moreover have (σ ⊗ *inv* τ) ⊗ (τ ⊗ *inv* ϱ) ∈ *U* **using** *that hV* **unfolding** *set-mult-def* **by** *fast*
ultimately show *?thesis* **using** *hU that* **by** *force*
qed
moreover have *?Φ ?F* **using** *hV* **by** *blast*
ultimately show *?thesis* **using** *hV* **by** *auto*
qed
moreover have ∀ *F*. *E* ⊆ *F* ∧ *F* ⊆ *carrier G* × *carrier G* ⟶ *?Φ F* **using** *hE* **by** *auto*
ultimately show *?thesis* **by** *blast*

qed
moreover have *?Φ* $(E \cap F)$ **if** *hEF*: *?Φ E* ∧ *?Φ F* **for** *E F*
proof −
from *hEF* **obtain** *U V* **where**
hU: *neighborhood* **1** *U* ∧ *right-entourage U* ⊆ *E* **and**
hV: *neighborhood* **1** *V* ∧ *right-entourage V* ⊆ *F*
by *presburger*
then have *neighborhood* **1** $(U \cap V)$ ∧ *right-entourage* $(U \cap V) \subseteq E \cap F$ **by** *fast*
then show *?thesis* **using** *that* **by** *auto*
qed
ultimately have *uniformity-on* (*carrier G*) *?Φ*
unfolding *uniformity-on-def* **by** *auto*
from *uniformity-inverse′*[*OF this*] **show** *?space-def* **and** *?entourage-def* **unfolding** *right-uniformity-def* **by** *auto*
qed

lemma *left-uniformity-induces-group-topology* [*simp*]:
shows *utopology left-uniformity* = *T*
proof −
let *?Φ* = *left-uniformity*
let *?T′* = *utopology ?Φ*
have *openin T U* ⟷ *openin ?T′ U* **for** *U*
proof
assume *U-open*: *openin T U*
have ∃ *E*. *entourage-in ?Φ E* ∧ *E*‘‘{σ} ⊆ *U* **if** *hσ*: σ ∈ *U* **for** σ
proof −
let *?E* = *left-entourage* (*inv* σ <# *U*)
have *in-group*: σ ∈ *carrier G* **using** *hσ U-open open-set-in-carrier* **by** *blast*
then have *openin T* (*inv* σ <# *U*)
using *inv-closed open-set-translations*(*1*) *U-open* **by** *presburger*
then have *neighborhood* **1** (*inv* σ <# *U*)
using *hσ in-group r-inv* **unfolding** *l-coset-def SET-INV-def* **by** *force*
then have *entourage-in ?Φ ?E* **unfolding** *entourage-left-uniformity* **by** *blast*
moreover have τ ∈ *U* **if** τ ∈ *?E*‘‘{σ} **for** τ
proof −
from *that* **have** *inv* σ ⊗ τ ∈ *inv* σ <# *U* **by** *force*
then obtain ϱ **where** *hϱ*: ϱ ∈ *U* ∧ *inv* σ ⊗ τ = *inv* σ ⊗ ϱ **unfolding** *l-coset-def* **by** *fast*
then have ϱ ∈ *carrier G* ∧ τ ∈ *carrier G* **using** *that open-set-in-carrier U-open* **by** *fast*
then have τ = ϱ **using** *in-group hϱ inv-closed* **by** (*metis Units-eq Units-l-cancel*)
then show *?thesis* **using** *hϱ* **by** *simp*
qed
ultimately show *?thesis* **by** *blast*
qed
moreover have *U* ⊆ *uspace ?Φ* **using** *openin-subset U-open* **by** *force*
ultimately show *openin ?T′ U* **unfolding** *openin-utopology* **by** *force*
next

assume *U-open*: *openin ?T′ U*
have ∃ *W*. *neighborhood σ W* ∧ *W* ⊆ *U* **if** *hσ*: *σ* ∈ *U* **for** *σ*
proof −
have *in-group*: *σ* ∈ *carrier G* **using** *hσ U-open openin-subset topspace-utopology* **by** *force*
from *U-open hσ* **obtain** *E* **where** *hE*: *entourage-in ?Φ E* ∧ *E“{σ}* ⊆ *U*
unfolding *openin-utopology* **by** *blast*
then obtain *V* **where** *hV*: *neighborhood* **1** *V* ∧ *left-entourage V* ⊆ *E*
unfolding *entourage-left-uniformity* **by** *fastforce*
let *?W* = {*τ* ∈ *carrier G*. *inv σ* ⊗ *τ* ∈ *V*}
from *hV* **have** *W-subset*: *?W* ⊆ *E“{σ}* **using** *in-group* **by** *fast*
have *continuous-map T T* (*λτ*. *inv σ* ⊗ *τ*) **using** *translations-continuous in-group inv-closed* **by** *blast*
then have *openin T ?W* **using** *openin-continuous-map-preimage hV* **by** *fastforce*
then have *neighborhood σ ?W* **using** *in-group r-inv hV* **by** *simp*
then show *?thesis* **using** *W-subset hE* **by** *fast*
qed
then show *openin T U* **using** *openin-subopen* **by** *force*
qed
then show *?thesis* **using** *topology-eq* **by** *blast*
qed

lemma *right-uniformity-induces-group-topology* [*simp*]:
shows *utopology right-uniformity* = *T*
proof −
let *?Φ* = *right-uniformity*
let *?T′* = *utopology ?Φ*
have *openin T U* ⟷ *openin ?T′ U* **for** *U*
proof
assume *U-open*: *openin T U*
have ∃ *E*. *entourage-in ?Φ E* ∧ *E“{σ}* ⊆ *U* **if** *hσ*: *σ* ∈ *U* **for** *σ*
proof −
let *?E* = *right-entourage* (*σ* <# *set-inv U*)
have *in-group*: *σ* ∈ *carrier G* **using** *hσ U-open open-set-in-carrier* **by** *blast*
then have *openin T* (*σ* <# *set-inv U*)
using *open-set-inv-open open-set-translations*(*1*) *U-open* **by** *presburger*
then have *neighborhood* **1** (*σ* <# *set-inv U*)
using *hσ in-group r-inv* **unfolding** *l-coset-def SET-INV-def* **by** *force*
then have *entourage-in ?Φ ?E* **unfolding** *entourage-right-uniformity* **by** *blast*
moreover have *τ* ∈ *U* **if** *τ* ∈ *?E“{σ}* **for** *τ*
proof −
from *that* **have** *σ* ⊗ *inv τ* ∈ *σ* <# *set-inv U* **by** *force*
then obtain *ϱ* **where** *hϱ*: *ϱ* ∈ *U* ∧ *σ* ⊗ *inv τ* = *σ* ⊗ *inv ϱ*
unfolding *l-coset-def SET-INV-def* **by** *fast*
then have *ϱ* ∈ *carrier G* ∧ *τ* ∈ *carrier G* **using** *that open-set-in-carrier U-open* **by** *fast*
then have *τ* = *ϱ* **using** *in-group hϱ inv-closed* **by** (*metis Units-eq Units-l-cancel inv-inv*)

then show *?thesis* **using** *h*ϱ **by** *simp*
qed
ultimately show *?thesis* **by** *blast*
qed
moreover have $U \subseteq$ *uspace ?*Φ **using** *openin-subset U-open* **by** *force*
ultimately show *openin ?T′ U* **unfolding** *openin-utopology* **by** *force*
next
assume *U-open*: *openin ?T′ U*
have $\exists$ *W. neighborhood* σ *W* $\wedge$ *W* $\subseteq$ *U* **if** *h*σ: $\sigma \in U$ **for** σ
proof −
have *in-group*: $\sigma \in$ *carrier G* **using** *h*σ *U-open openin-subset topspace-utopology* **by** *force*
from *U-open h*σ **obtain** *E* **where** *hE*: *entourage-in ?*Φ *E* $\wedge$ *E‘‘*$\{\sigma\} \subseteq U$
unfolding *openin-utopology* **by** *blast*
then obtain *V* **where** *hV*: *neighborhood* **1** *V* $\wedge$ *right-entourage V* $\subseteq$ *E*
unfolding *entourage-right-uniformity* **by** *fastforce*
let *?W* = $\{\tau \in$ *carrier G*. $\sigma \otimes$ *inv* $\tau \in V\}$
from *hV* **have** *W-subset*: *?W* $\subseteq$ *E‘‘*$\{\sigma\}$ **using** *in-group* **by** *fast*
have $(\lambda\tau.\ \sigma \otimes$ *inv* $\tau) = (\lambda\tau.\ \sigma \otimes \tau) \circ (\lambda\tau.$ *inv* $\tau)$ **by** *fastforce*
then have *continuous-map T T* $(\lambda\tau.\ \sigma \otimes$ *inv* $\tau)$ **using** *continuous-map-compose inv-continuous*
translations-continuous[*OF in-group*] **by** *metis*
then have *openin T ?W* **using** *openin-continuous-map-preimage hV* **by** *fastforce*
then have *neighborhood* σ *?W* **using** *in-group r-inv hV* **by** *simp*
then show *?thesis* **using** *W-subset hE* **by** *fast*
qed
then show *openin T U* **using** *openin-subopen* **by** *force*
qed
then show *?thesis* **using** *topology-eq* **by** *blast*
qed

lemma *translations-ucontinuous*:
assumes *in-group*: $\sigma \in$ *carrier G*
shows *ucontinuous left-uniformity left-uniformity* $(\lambda\tau.\ \sigma \otimes \tau)$ **and**
ucontinuous right-uniformity right-uniformity $(\lambda\tau.\ \tau \otimes \sigma)$
proof −
let *?*Φ = *left-uniformity*
have *entourage-in ?*Φ $\{(\tau_1, \tau_2) \in$ *uspace ?*$\Phi \times$ *uspace ?*Φ. $(\sigma \otimes \tau_1, \sigma \otimes \tau_2) \in E\}$
if *hE*: *entourage-in ?*Φ *E* **for** *E*
proof −
let *?F* = $\{(\tau_1, \tau_2) \in$ *uspace ?*$\Phi \times$ *uspace ?*Φ. $(\sigma \otimes \tau_1, \sigma \otimes \tau_2) \in E\}$
from *hE* **obtain** *U* **where** *hU*: *neighborhood* **1** *U* $\wedge$ *left-entourage U* $\subseteq$ *E*
unfolding *entourage-left-uniformity* **by** *auto*
have $(\tau_1, \tau_2) \in$ *?F* **if** $\tau_1 \in$ *carrier G* $\wedge$ $\tau_2 \in$ *carrier G* $\wedge$ *inv* $\tau_1 \otimes \tau_2 \in U$ **for** $\tau_1\ \tau_2$
proof −
have *inv* $(\sigma \otimes \tau_1) \otimes (\sigma \otimes \tau_2) =$ *inv* $\tau_1 \otimes \tau_2$

using *that in-group m-closed inv-closed inv-mult-group m-assoc r-inv r-one*
by (*smt* (*verit*, *ccfv-threshold*))
then have $(\sigma \otimes \tau_1, \sigma \otimes \tau_2) \in E$ **using** *that hU in-group m-closed* **by** *fastforce*
then show *?thesis* **using** *that* **by** *auto*
qed
then have *left-entourage* $U \subseteq$ *?F* **by** *force*
then show *?thesis* **unfolding** *entourage-left-uniformity* **using** *hU* **by** *auto*
qed
moreover have $(\lambda\tau.\ \sigma \otimes \tau) \in$ *uspace ?Φ* $\rightarrow$ *uspace ?Φ*
unfolding *Pi-def* **using** *uspace-left-uniformity in-group m-closed* **by** *force*
ultimately show *ucontinuous ?Φ ?Φ* $(\lambda\tau.\ \sigma \otimes \tau)$
unfolding *ucontinuous-def* **by** *fast*
next
let *?Φ* = *right-uniformity*
have *entourage-in ?Φ* $\{(\tau_1, \tau_2) \in$ *uspace ?Φ* $\times$ *uspace ?Φ*. $(\tau_1 \otimes \sigma, \tau_2 \otimes \sigma) \in E\}$
if *hE*: *entourage-in ?Φ E* **for** *E*
proof −
let *?F* = $\{(\tau_1, \tau_2) \in$ *uspace ?Φ* $\times$ *uspace ?Φ*. $(\tau_1 \otimes \sigma, \tau_2 \otimes \sigma) \in E\}$
from *hE* **obtain** *U* **where** *hU*: *neighborhood* **1** $U \land$ *right-entourage* $U \subseteq E$
unfolding *entourage-right-uniformity* **by** *auto*
have $(\tau_1, \tau_2) \in$ *?F* **if** $\tau_1 \in$ *carrier G* $\land\ \tau_2 \in$ *carrier G* $\land\ \tau_1 \otimes$ *inv* $\tau_2 \in U$ **for** $\tau_1\ \tau_2$
proof −
have $(\tau_1 \otimes \sigma) \otimes$ *inv* $(\tau_2 \otimes \sigma) = \tau_1 \otimes$ *inv* τ_2
using *that in-group m-closed inv-closed inv-mult-group m-assoc r-inv r-one*
by (*smt* (*verit*, *ccfv-threshold*))
then have $(\tau_1 \otimes \sigma, \tau_2 \otimes \sigma) \in E$ **using** *that hU in-group m-closed* **by** *fastforce*
then show *?thesis* **using** *that* **by** *simp*
qed
then have *right-entourage* $U \subseteq$ *?F* **by** *force*
then show *?thesis* **unfolding** *entourage-right-uniformity* **using** *hU* **by** *auto*
qed
moreover have $(\lambda\tau.\ \tau \otimes \sigma) \in$ *uspace ?Φ* $\rightarrow$ *uspace ?Φ*
unfolding *Pi-def* **using** *entourage-right-uniformity in-group m-closed* **by** *force*
ultimately show *ucontinuous ?Φ ?Φ* $(\lambda\tau.\ \tau \otimes \sigma)$
unfolding *ucontinuous-def* **by** *fast*
qed

2.5 The Birkhoff-Kakutani theorem

2.5.1 Prenorms on groups

definition *group-prenorm* :: $('g \Rightarrow real) \Rightarrow bool$ **where**
group-prenorm $N \longleftrightarrow$
$N\ \mathbf{1} = 0\ \land$
$(\forall \sigma\ \tau.\ \sigma \in$ *carrier G* $\land\ \tau \in$ *carrier G* $\longrightarrow N\ (\sigma \otimes \tau) \le N\ \sigma + N\ \tau) \land$
$(\forall \sigma \in$ *carrier G*. N (*inv* σ) $= N\ \sigma)$

lemma *group-prenorm-clauses*[*elim*]:

assumes *group-prenorm N*
obtains
$N\ \mathbf{1} = 0$ **and**
$\bigwedge\sigma\ \tau.\ \sigma \in carrier\ G \Longrightarrow \tau \in carrier\ G \Longrightarrow N\ (\sigma \otimes \tau) \leq N\ \sigma + N\ \tau$ **and**
$\bigwedge\sigma.\ \sigma \in carrier\ G \Longrightarrow N\ (inv\ \sigma) = N\ \sigma$
using *assms* **unfolding** *group-prenorm-def* **by** *auto*

proposition *group-prenorm-nonnegative*:
assumes *prenorm*: *group-prenorm N*
shows $\forall\sigma \in carrier\ G.\ N\ \sigma \geq 0$
proof
fix σ **assume** $\sigma \in carrier\ G$
from *r-inv this* **have** $0 \leq N\ \sigma + N\ \sigma$ **using** *assms inv-closed group-prenorm-clauses* **by** *metis*
then show $N\ \sigma \geq 0$ **by** *fastforce*
qed

proposition *group-prenorm-reverse-triangle-ineq*:
assumes *prenorm*: *group-prenorm N* **and** *in-group*: $\sigma \in carrier\ G \wedge \tau \in carrier\ G$
shows $|N\ \sigma - N\ \tau| \leq N\ (\sigma \otimes inv\ \tau)$
proof −
have $\sigma = \sigma \otimes inv\ \tau \otimes \tau$ **using** *in-group inv-closed r-one l-inv m-assoc* **by** *metis*
then have *a*: $N\ \sigma \leq N\ (\sigma \otimes inv\ \tau) + N\ \tau$ **using** *in-group inv-closed m-closed prenorm group-prenorm-clauses* **by** *metis*
have $inv\ \tau = inv\ \sigma \otimes (\sigma \otimes inv\ \tau)$ **using** *in-group inv-closed l-one l-inv m-assoc* **by** *metis*
then have *b*: $N\ \tau \leq N\ \sigma + N\ (\sigma \otimes inv\ \tau)$ **using** *in-group inv-closed m-closed prenorm group-prenorm-clauses* **by** *metis*
from *a b* **show** *?thesis* **by** *linarith*
qed

definition *induced-group-prenorm* :: $('g \Rightarrow real) \Rightarrow 'g \Rightarrow real$ **where**
$induced\text{-}group\text{-}prenorm\ f\ \sigma = (SUP\ \tau \in carrier\ G.\ |f\ (\tau \otimes \sigma) - f\ \tau|)$

lemma *induced-group-prenorm-welldefined*:
fixes $f :: 'g \Rightarrow real$
assumes *f-bounded*: $\exists\, c.\forall\tau \in carrier\ G.\ |f\ \tau| \leq c$ **and** *in-group*: $\sigma \in carrier\ G$
shows *bdd-above* $((\lambda\tau.\ |f\ (\tau \otimes \sigma) - f\ \tau|)\,`(carrier\ G))$
proof −
from *f-bounded* **obtain** *c* **where** *hc*: $\forall\tau \in carrier\ G.\ |f\ \tau| \leq c$ **by** *blast*
have $|f\ (\tau \otimes \sigma) - f\ \tau| \leq 2{*}c$ **if** $\tau \in carrier\ G$ **for** τ
proof −
have $|f\ (\tau \otimes \sigma) - f\ \tau| \leq |f\ (\tau \otimes \sigma)| + |f\ \tau|$ **using** *abs-triangle-ineq* **by** *simp*
then show *?thesis* **using** *in-group that m-closed f-bounded hc* **by** (*smt* (*verit*, *best*))
qed
then show *?thesis* **unfolding** *bdd-above-def image-def* **by** *blast*
qed

lemma *bounded-function-induces-group-prenorm*:
fixes f :: $'g \Rightarrow real$
assumes *f-bounded*: $\exists\, c. \forall\, \sigma \in$ *carrier G*. $|f\ \sigma| \leq c$
shows *group-prenorm* (*induced-group-prenorm f*)
proof −
let $?N = \lambda\sigma.$ *SUP* $\tau \in$ *carrier G*. $|f\ (\tau \otimes \sigma) - f\ \tau|$
have $?N\ \mathbf{1} =$ (*SUP* $\tau \in$ *carrier G*. *0*) **using** *r-one* **by** *simp*
then have $?N\ \mathbf{1} = 0$ **using** *carrier-not-empty* **by** *simp*
moreover have $?N\ (\sigma \otimes \tau) \leq ?N\ \sigma + ?N\ \tau$ **if** $h\sigma\tau$: $\sigma \in$ *carrier G* $\wedge\ \tau \in$ *carrier G* **for** $\sigma\ \tau$
proof −
have $|f\ (\varrho \otimes (\sigma \otimes \tau)) - f\ \varrho| \leq ?N\ \sigma + ?N\ \tau$ **if** $\varrho \in$ *carrier G* **for** ϱ
proof −
have *a*: $|f\ (\varrho \otimes (\sigma \otimes \tau)) - f\ \varrho| \leq |f\ (\varrho \otimes (\sigma \otimes \tau)) - f\ (\varrho \otimes \sigma)| + |f\ (\varrho \otimes \sigma) - f\ \varrho|$
using *abs-triangle-ineq* **by** *linarith*
have $|f\ (\varrho \otimes \sigma \otimes \tau) - f\ (\varrho \otimes \sigma)| \leq ?N\ \tau$
using *induced-group-prenorm-welldefined*[*OF f-bounded*] *that* $h\sigma\tau$ *m-closed cSUP-upper* **by** *meson*
then have *b*: $|f\ (\varrho \otimes (\sigma \otimes \tau)) - f\ (\varrho \otimes \sigma)| \leq ?N\ \tau$ **using** *m-assoc that* $h\sigma\tau$ **by** *simp*
have *c*: $|f\ (\varrho \otimes \sigma) - f\ \varrho| \leq ?N\ \sigma$ **using** *induced-group-prenorm-welldefined*[*OF f-bounded*] $h\sigma\tau$ *that cSUP-upper* **by** *meson*
from *a b c* **show** *?thesis* **by** *argo*
qed
then show *?thesis* **using** *cSUP-least carrier-not-empty* **by** *meson*
qed
moreover have $?N\ (inv\ \sigma) = ?N\ \sigma$ **if** $h\sigma$: $\sigma \in$ *carrier G* **for** σ
proof −
have $|f\ (\tau \otimes inv\ \sigma) - f\ \tau| \in \{|f\ (\varrho \otimes \sigma) - f\ \varrho| \mid \varrho.\ \varrho \in$ *carrier G* $\}$ **if** $\tau \in$ *carrier G* **for** τ
proof −
have $|f\ (\tau \otimes inv\ \sigma) - f\ \tau| = |f\ (\tau \otimes inv\ \sigma) - f\ (\tau \otimes inv\ \sigma \otimes \sigma)|$
using $h\sigma$ *that m-assoc r-one l-inv* **by** *simp*
then have $|f\ (\tau \otimes inv\ \sigma) - f\ \tau| = |f\ (\tau \otimes inv\ \sigma \otimes \sigma) - f\ (\tau \otimes inv\ \sigma)|$ **by** *argo*
then show *?thesis* **using** $h\sigma$ *that m-closed* **by** *blast*
qed
moreover
have $|f\ (\varrho \otimes \sigma) - f\ \varrho| \in \{|f\ (\tau \otimes inv\ \sigma) - f\ \tau| \mid \tau.\ \tau \in$ *carrier G* $\}$ **if** $\varrho \in$ *carrier G* **for** ϱ
proof −
have $|f\ (\varrho \otimes \sigma) - f\ \varrho| = |f\ (\varrho \otimes \sigma) - f\ (\varrho \otimes \sigma \otimes inv\ \sigma)|$
using $h\sigma$ *that m-assoc r-one r-inv* **by** *simp*
then have $|f\ (\varrho \otimes \sigma) - f\ \varrho| = |f\ (\varrho \otimes \sigma \otimes inv\ \sigma) - f\ (\varrho \otimes \sigma)|$ **by** *argo*
then show *?thesis* **using** $h\sigma$ *that* **by** *blast*
qed
ultimately have $\{|f\ (\tau \otimes inv\ \sigma) - f\ \tau| \mid \tau.\ \tau \in$ *carrier G*$\} = \{|f\ (\varrho \otimes \sigma) -$

f ϱ| | *ϱ*. *ϱ* ∈ *carrier G*} **by** *blast*
  then show *?thesis* **by** (*simp add*: *setcompr-eq-image*)
 qed
 ultimately show *?thesis* **unfolding** *induced-group-prenorm-def group-prenorm-def*
by *fast*
qed

lemma *neighborhood-1-translation*:
 assumes *neighborhood* **1** *U* **and** *σ* ∈ *carrier G* ∨ *σ* ∈ *topspace T*
 shows *neighborhood σ* (*σ* <# *U*)
proof −
 have *openin T* (*σ* <# *U*) **using** *assms open-set-translations*(*1*) **by** *simp*
 then show *?thesis* **unfolding** *l-coset-def* **using** *assms r-one* **by** *force*
qed

proposition *group-prenorm-continuous-if-continuous-at-1*:
 assumes *prenorm*: *group-prenorm N* **and**
  continuous-at-1: ∀ *ε*>*0*.∃ *U*. *neighborhood* **1** *U* ∧ (∀ *σ*∈*U*. *N σ* < *ε*)
 shows *continuous-map T euclideanreal N*
proof −
 have ∃ *V*. *neighborhood σ V* ∧ (∀ *τ*∈*V*. *N τ* ∈ *Met-TC.mball* (*N σ*) *ε*)
  if *hσ*: *σ* ∈ *topspace T* **and** *hε*: *ε* > *0* **for** *σ ε*
 proof −
  from *continuous-at-1* **obtain** *U* **where** *hU*: *neighborhood* **1** *U* ∧ (∀ *τ*∈*U*. *N τ*
< *ε*) **using** *hε* **by** *presburger*
  then have *neighborhood σ* (*σ* <# *U*) **using** *hσ neighborhood-1-translation* **by**
blast
  moreover have *N* (*σ* ⊗ *τ*) ∈ *Met-TC.mball* (*N σ*) *ε* **if** *τ* ∈ *U* **for** *τ*
  proof −
   have *in-group*: *σ* ∈ *carrier G* ∧ *τ* ∈ *carrier G* **using** *hσ that openin-subset*
hU **by** *blast*
   then have (*inv σ*) ⊗ (*σ* ⊗ *τ*) = *τ* **using** *l-inv l-one m-assoc inv-closed* **by**
metis
   then have |*N* (*inv σ*) − *N* (*inv* (*σ* ⊗ *τ*))| ≤ *N τ* **using** *group-prenorm-reverse-triangle-ineq*

    in-group inv-closed m-closed **by** (*metis inv-inv prenorm*)
   then have |*N σ* − *N* (*σ* ⊗ *τ*)| < *ε*
    using *prenorm in-group m-closed inv-closed hU that* **by** *fastforce*
   then show *?thesis* **unfolding** *Met-TC.mball-def dist-real-def* **by** *fast*
  qed
  ultimately show *?thesis* **unfolding** *l-coset-def* **by** *blast*
 qed
 then show *?thesis* **using** *Metric-space.continuous-map-to-metric*
  by (*metis Met-TC.Metric-space-axioms mtopology-is-euclidean*)
qed

2.5.2 A prenorm respecting the group topology

context

```
  fixes U :: nat ⇒ 'g set
  assumes U-neighborhood: ∀ n. neighborhood 1 (U n)
  assumes U-props: ∀ n. symmetric (U n) ∧ (U (n + 1)) <#> (U (n + 1)) ⊆
(U n)
begin

private fun V :: nat ⇒ nat ⇒ 'g set where
V m n = (
  if m = 0 then {} else
  if m = 1 then U n else
  if m > 2^n then carrier G else
  if even m then V (m div 2) (n − 1) else
  V ((m − 1) div 2) (n − 1) <#> U n
)

private lemma U-in-group: U k ⊆ carrier G using U-neighborhood open-set-in-carrier
by fast

private lemma V-in-group:
  shows V m n ⊆ carrier G
proof (induction n arbitrary: m)
  case (Suc n)
  then have V ((m − 1) div 2) n <#> U (Suc n) ⊆ carrier G
    unfolding set-mult-def using U-in-group by fast
  then show ?case using U-in-group Suc by simp
qed (auto simp: U-in-group)

private lemma V-mult:
  shows m ≥ 1 ⟹ V m n <#> U n ⊆ V (m + 1) n
proof (induction n arbitrary: m)
  case 0
  then have V (m + 1) 0 = carrier G by simp
  then show ?case unfolding set-mult-def using V-in-group U-in-group by fast
next
  case (Suc n)
  then show ?case
  proof (cases m + 1 > 2^(Suc n))
    case True
    then have V (m + 1) (Suc n) = carrier G by force
     then show ?thesis unfolding set-mult-def using V-in-group U-in-group by
blast
  next
    case m-in-bounds: False
    then show ?thesis
    proof (cases m = 1)
      case True
      then show ?thesis using U-in-group U-props by force
    next
      case m-not-1: False
```

then show *?thesis*
proof (*cases even m*)
case *True*
then have *V m* (*Suc n*) <#> *U* (*Suc n*) = *V* (*m* + *1*) (*Suc n*) **using** *m-in-bounds m-not-1 Suc*(*2*) **by** *auto*
then show *?thesis* **by** *blast*
next
case *m-odd*: *False*
have *U-mult*: *U* (*Suc n*) <#> *U* (*Suc n*) ⊆ *U n* **using** *U-props* **by** *simp*
have *not-zero*: (*m* − *1*) *div 2* ≥ *1* **using** *Suc*(*2*) *m-not-1 m-odd* **by** *presburger*
have *arith*: (*m* − *1*) *div 2* + *1* = (*m* + *1*) *div 2* **using** *Suc*(*2*) **by** *simp*
have *V m* (*Suc n*) <#> *U* (*Suc n*) = *V* ((*m* − *1*) *div 2*) *n* <#> *U* (*Suc n*) <#> *U* (*Suc n*) **using** *m-odd m-in-bounds m-not-1 Suc*(*2*) **by** *simp*
also have ... = *V* ((*m* − *1*) *div 2*) *n* <#> (*U* (*Suc n*) <#> *U* (*Suc n*)) **using** *set-mult-assoc V-in-group U-in-group* **by** *simp*
also have ... ⊆ *V* ((*m* − *1*) *div 2*) *n* <#> *U n* **using** *mono-set-mult U-mult* **by** *blast*
also have ... ⊆ *V* ((*m* − *1*) *div 2* + *1*) *n* **using** *Suc*(*1*) *not-zero* **by** *blast*
also have ... = *V* ((*m* + *1*) *div 2*) *n* **using** *arith* **by** *presburger*
also have ... = *V* (*m* + *1*) (*Suc n*) **using** *m-odd m-not-1 m-in-bounds Suc*(*2*) **by** *simp*
finally show *?thesis* **by** *blast*
qed
qed
qed
qed

private lemma *V-mono*:
assumes *smaller*: (*real* m_1)/*2*^n_1 ≤ (*real* m_2)/*2*^n_2 **and** *not-zero*: $m_1 \geq 1 \wedge m_2 \geq 1$
shows *V* m_1 n_1 ⊆ *V* m_2 n_2
proof −
have *V m n* ⊆ *V* (*m* + *1*) *n* **if** *m* ≥ *1* **for** *m n*
proof −
have *V m n* <#> *U n* ⊆ *V* (*m* + *1*) *n* **using** *V-mult U-props that* **by** *presburger*
moreover have *V m n* ⊆ *carrier G* ∧ *U n* ⊆ *carrier G* **using** *U-in-group V-in-group* **by** *auto*
ultimately show *?thesis* **using** *set-mult-one-subset U-neighborhood* **by** *blast*
qed
then have *subset-suc*: *V m n* ⊆ *V* (*m* + *1*) *n* **for** *m n* **by** *simp*
have *V m n* ⊆ *V* (*m* + *k*) *n* **for** *m n k*
proof (*induction k*)
case (*Suc k*)
then show *?case* **unfolding** *Suc-eq-plus1* **using** *subset-suc Suc*
by (*metis* (*no-types*, *opaque-lifting*) *add.assoc dual-order.trans*)
qed (*simp*)
then have *a*: *V m n* ⊆ *V m′ n* **if** *m′* ≥ *m* **for** *m m′ n* **using** *that le-Suc-ex* **by** *blast*
have *b*: *V m n* = *V* (*m* ∗ *2*^*k*) (*n*+*k*) **if** *m* ≥ *1* **for** *m n k*

proof (*induction k*)
case (*Suc k*)
have $V\ (m * 2\hat{}k * 2)\ (n + k + 1) = V\ (m * 2\hat{}k)\ (n + k)$ **using** *that* **by** *simp*
then show *?case* **unfolding** *Suc-eq-plus1* **using** *Suc* **by** *simp*
qed (*auto*)
show *?thesis*
proof (*cases* $n_1 \leq n_2$)
case *True*
have $(real\ m_1)/2\hat{}n_1 = (real\ (m_1 * 2\hat{}(n_2 - n_1)))/(2\hat{}n_1 * 2\hat{}(n_2 - n_1))$ **by** *fastforce*
also have ... $= (real\ (m_1 * 2\hat{}(n_2 - n_1)))/2\hat{}n_2$ **using** *True* **by** (*metis le-add-diff-inverse power-add*)
finally have $(real\ (m_1 * 2\hat{}(n_2 - n_1)))/2\hat{}n_2 \leq (real\ m_2)/2\hat{}n_2$ **using** *smaller* **by** *fastforce*
then have *ineq*: $m_1 * 2\hat{}(n_2 - n_1) \leq m_2$
by (*smt* (*verit*) *divide-cancel-right divide-right-mono linorder-le-cases of-nat-eq-iff of-nat-mono order-antisym-conv power-not-zero zero-le-power*)
from *b* **have** $V\ m_1\ n_1 = V\ (m_1 * 2\hat{}(n_2 - n_1))\ (n_1 + (n_2 - n_1))$ **using** *not-zero* **by** *blast*
also have ... $= V\ (m_1 * 2\hat{}(n_2 - n_1))\ n_2$ **using** *True* **by** *force*
finally show *?thesis* **using** *a*[*OF ineq*] **by** *blast*
next
case *False*
then have n_2*-leq-*n_1: $n_2 \leq n_1$ **by** *simp*
have $(real\ m_2)/2\hat{}n_2 = (real\ (m_2 * 2\hat{}(n_1 - n_2)))/(2\hat{}n_2 * 2\hat{}(n_1 - n_2))$ **by** *fastforce*
also have ... $= (real\ (m_2 * 2\hat{}(n_1 - n_2)))/2\hat{}n_1$ **using** n_2*-leq-*n_1 **by** (*metis le-add-diff-inverse power-add*)
finally have $(real\ (m_2 * 2\hat{}(n_1 - n_2)))/2\hat{}n_1 \geq (real\ m_1)/2\hat{}n_1$ **using** *smaller* **by** *fastforce*
then have *ineq*: $m_2 * 2\hat{}(n_1 - n_2) \geq m_1$
by (*smt* (*verit*) *divide-cancel-right divide-right-mono linorder-le-cases of-nat-eq-iff of-nat-mono order-antisym-conv power-not-zero zero-le-power*)
from *b* **have** $V\ m_2\ n_2 = V\ (m_2 * 2\hat{}(n_1 - n_2))\ (n_2 + (n_1 - n_2))$ **using** *not-zero* **by** *blast*
also have ... $= V\ (m_2 * 2\hat{}(n_1 - n_2))\ n_1$ **using** n_2*-leq-*n_1 **by** *force*
finally show *?thesis* **using** *a*[*OF ineq*] **by** *blast*
qed
qed

private lemma *approx-number-by-multiples*:
assumes *hx*: $x \geq 0$ **and** *hc*: $c > 0$
shows $\exists\, k :: nat \geq 1.\ (real\ (k-1))/c \leq x \wedge x < (real\ k)/c$
proof −
let $?k = \lfloor x * c \rfloor + 1$
have $?k \geq 1$ **using** *assms* **by** *simp*
moreover from *this* **have** $real\ (nat\ ?k) = ?k$ **by** *auto*
moreover have $(?k-1)/c \leq x \wedge x < ?k/c$

using *assms* **by** (*simp add*: *mult-imp-div-pos-le pos-less-divide-eq*)
ultimately show *?thesis*
by (*smt* (*verit*) *nat-diff-distrib nat-le-eq-zle nat-one-as-int of-nat-nat*)
qed

lemma *construction-of-prenorm-respecting-topology*:
shows $\exists N.$ *group-prenorm* $N \wedge$
$(\forall n. \{\sigma \in carrier\ G.\ N\ \sigma < 1/2\hat{}n\} \subseteq U\ n) \wedge$
$(\forall n.\ U\ n \subseteq \{\sigma \in carrier\ G.\ N\ \sigma \leq 2/2\hat{}n\})$
proof −
define $f :: 'g \Rightarrow real$ **where** $f\ \sigma = Inf\ \{(real\ m)/2\hat{}n \mid m\ n.\ \sigma \in V\ m\ n\}$ **for** σ
define $N :: 'g \Rightarrow real$ **where** $N =$ *induced-group-prenorm f*

have $\sigma \in V\ 2\ 0$ **if** $\sigma \in carrier\ G$ **for** σ **using** *that* **by** *auto*
then have *contains-2*: $(real\ 2)/2\hat{}0 \in \{(real\ m)/2\hat{}n \mid m\ n.\ \sigma \in V\ m\ n\}$ **if** $\sigma \in carrier\ G$ **for** σ **using** *that* **by** *blast*
then have *nonempty*: $\{(real\ m)/2\hat{}n \mid m\ n.\ \sigma \in V\ m\ n\} \neq \{\}$ **if** $\sigma \in carrier\ G$ **for** σ **using** *that* **by** *fast*
have *positive*: $(real\ m)/2\hat{}n \geq 0$ **for** $m\ n$ **by** *simp*
then have *bdd-below*: *bdd-below* $\{(real\ m)/2\hat{}n \mid m\ n.\ \sigma \in V\ m\ n\}$ **for** σ **by** *fast*
have *f-bounds*: $0 \leq f\ \sigma \wedge f\ \sigma \leq 2$ **if** $h\sigma$: $\sigma \in carrier\ G$ **for** σ
proof −
from *bdd-below* **have** $f\ \sigma \leq (real\ 2)/2\hat{}0$ **unfolding** *f-def* **using** *cInf-lower contains-2*[*OF h*σ] **by** *meson*
moreover have $0 \leq f\ \sigma$ **using** *cInf-greatest contains-2*[*OF h*σ] **unfolding** *f-def* **using** *positive*
by (*smt* (*verit*, *del-insts*) *Collect-mem-eq empty-Collect-eq mem-Collect-eq*)
ultimately show *?thesis* **by** *fastforce*
qed
then have *N-welldefined*: *bdd-above* $((\lambda\tau.\ |f\ (\tau \otimes \sigma) - f\ \tau|)$ ‘ *carrier G*) **if** $\sigma \in carrier\ G$ **for** σ
using *induced-group-prenorm-welldefined that* **by** (*metis* (*full-types*) *abs-of-nonneg*)

have *in-V-if-f-smaller*: $\sigma \in V\ m\ n$ **if** $h\sigma$: $\sigma \in carrier\ G$ **and** *smaller*: $f\ \sigma < (real\ m)/2\hat{}n$ **for** $\sigma\ m\ n$
proof −
from *cInf-lessD* **obtain** q **where** *hq*: $q \in \{(real\ m)/2\hat{}n \mid m\ n.\ \sigma \in V\ m\ n\} \wedge q < (real\ m)/2\hat{}n$
using *smaller nonempty*[*OF h*σ] **unfolding** *f-def* **by** (*metis* (*mono-tags*, *lifting*))
then obtain $m'\ n'$ **where** $hm'n'$: $\sigma \in V\ m'\ n' \wedge q = (real\ m')/2\hat{}n'$ **by** *fast*
moreover have $m' \geq 1$
proof (*rule ccontr*)
assume $\neg\ m' \geq 1$
then have $V\ m'\ n' = \{\}$ **by** *force*
then show *False* **using** $hm'n'$ **by** *blast*
qed
moreover have $m \geq 1$ **using** *f-bounds smaller h*σ
by (*metis divide-eq-0-iff less-numeral-extra*(*3*) *less-one linorder-le-less-linear*

nle-le of-nat-0 order-less-imp-le)
ultimately have *V m′ n′* ⊆ *V m n* **using** *V-mono hq U-props open-set-in-carrier* **by** *simp*
then show *?thesis* **using** *hm′n′* **by** *fast*
qed
have *f-1-vanishes*: *f* **1** = *0*
proof (*rule ccontr*)
assume *f* **1** ≠ *0*
then have *f* **1** > *0* **using** *f-bounds* **by** *fastforce*
then obtain *n* **where** *hn*: *f* **1** > (*real 1*)/*2*^*n*
by (*metis divide-less-eq-1 of-nat-1 one-less-numeral-iff power-one-over real-arch-pow-inv semiring-norm*(*76*) *zero-less-numeral*)
have **1** ∈ *V 1 n* **using** *U-neighborhood* **by** *simp*
then have (*real 1*)/*2*^*n* ∈ {(*real m*)/*2*^*n* |*m n*. **1** ∈ *V m n*} **by** *fast*
then show *False* **using** *hn cInf-lower bdd-below*[*of* **1**] **unfolding** *f-def* **by** (*smt* (*verit*, *ccfv-threshold*))
qed
have *in-U-if-N-small*: σ ∈ *U n* **if** *in-group*: σ ∈ *carrier G* **and** *N-small*: *N* σ < *1*/*2*^*n* **for** σ *n*
proof −
have *f* σ = |*f* (**1** ⊗ σ) − *f* **1**| **using** *in-group l-one f-1-vanishes f-bounds* **by** *force*
moreover have ... ≤ *N* σ **unfolding** *N-def induced-group-prenorm-def*
using *cSUP-upper N-welldefined*[*OF in-group*] **by** (*metis* (*mono-tags*, *lifting*) *one-closed*)
ultimately have σ ∈ *V 1 n* **using** *in-V-if-f-smaller*[*OF in-group*] *N-small* **by** (*smt* (*verit*) *of-nat-1*)
then show *?thesis* **by** *fastforce*
qed
have *N-bounds*: *N* σ ≤ *2*/*2*^*n* **if** *h*σ: σ ∈ *U n* **for** σ *n*
proof −
have *diff-bounded*: *f* (τ ⊗ σ) − *f* τ ≤ *2*/*2*^*n* ∧ *f* (τ ⊗ *inv* σ) − *f* τ ≤ *2*/*2*^*n* **if** *h*τ: τ ∈ *carrier G* **for** τ
proof −
obtain *k* **where** *hk*: *k* ≥ *1* ∧ (*real* (*k*−*1*))/*2*^*n* ≤ *f* τ ∧ *f* τ < (*real k*)/*2*^*n*
using *approx-number-by-multiples*[*of f* τ *2*^*n*] *f-bounds*[*OF h*τ] **by** *auto*
then have τ ∈ *V k n* **using** *in-V-if-f-smaller*[*OF h*τ] **by** *blast*
moreover have σ ∈ *V 1 n* ∧ *inv* σ ∈ *V 1 n* **using** *h*σ *U-props* **by** *auto*
moreover have *V k n* <#> *V 1 n* ⊆ *V* (*k* + *1*) *n*
using *V-mult U-props open-set-in-carrier hk* **by** *auto*
ultimately have τ ⊗ σ ∈ *V* (*k* + *1*) *n* ∧ τ ⊗ *inv* σ ∈ *V* (*k* + *1*) *n*
unfolding *set-mult-def* **by** *fast*
then have *a*: (*real* (*k* + *1*))/*2*^*n* ∈ {(*real m*)/*2*^*n* | *m n*. τ ⊗ σ ∈ *V m n*}
∧ (*real* (*k* + *1*))/*2*^*n* ∈ {(*real m*)/*2*^*n* | *m n*. τ ⊗ *inv* σ ∈ *V m n*} **by** *fast*
then have *f* (τ ⊗ σ) ≤ (*real* (*k* + *1*))/*2*^*n*
unfolding *f-def* **using** *cInf-lower*[*of* (*real* (*k* + *1*))/*2*^*n*] *bdd-below* **by** *presburger*
moreover from *a* **have** *f* (τ ⊗ *inv* σ) ≤ (*real* (*k* + *1*))/*2*^*n*
unfolding *f-def* **using** *cInf-lower*[*of* (*real* (*k* + *1*))/*2*^*n*] *bdd-below* **by**

presburger
ultimately show *?thesis* **using** *hk*
by (*smt* (*verit*, *ccfv-SIG*) *diff-divide-distrib of-nat-1 of-nat-add of-nat-diff*)
qed
have $|f\ (\varrho \otimes \sigma) - f\ \varrho| \leq 2/2\hat{}n$ **if** $h\varrho$: $\varrho \in carrier\ G$ **for** ϱ
proof −
have *in-group*: $\sigma \in carrier\ G$ **using** $h\sigma$ *U-in-group* **by** *fast*
then have $f\ (\varrho \otimes \sigma \otimes inv\ \sigma) - f\ (\varrho \otimes \sigma) \leq 2/2\hat{}n$ **using** *diff-bounded*[*of* $\varrho \otimes \sigma$] $h\varrho$ *m-closed* **by** *fast*
moreover have $\varrho \otimes \sigma \otimes inv\ \sigma = \varrho$ **using** *m-assoc r-inv r-one in-group inv-closed* $h\varrho$ **by** *presburger*
ultimately have $f\ \varrho - f\ (\varrho \otimes \sigma) \leq 2/2\hat{}n$ **by** *force*
moreover have $f\ (\varrho \otimes \sigma) - f\ \varrho \leq 2/2\hat{}n$ **using** *diff-bounded*[*OF* $h\varrho$] **by** *fast*
ultimately show *?thesis* **by** *force*
qed
then show *?thesis* **unfolding** *N-def induced-group-prenorm-def* **using** *cSUP-least carrier-not-empty* **by** *meson*
qed
then have $U\ n \subseteq \{\sigma \in carrier\ G.\ N\ \sigma \leq 2/2\hat{}n\}$ **for** n **using** *U-in-group* **by** *blast*
moreover have *group-prenorm N* **unfolding** *N-def*
using *bounded-function-induces-group-prenorm f-bounds* **by** (*metis abs-of-nonneg*)
ultimately show *?thesis* **using** *in-U-if-N-small* **by** *blast*
qed
end

2.5.3 Proof of Birkhoff-Kakutani

lemma *first-countable-neighborhoods-of-1-sequence*:
assumes *first-countable T*
shows $\exists\ U :: nat \Rightarrow {}'g\ set.$
$(\forall n.\ neighborhood\ \mathbf{1}\ (U\ n) \wedge symmetric\ (U\ n) \wedge U\ (n+1) <\#> U\ (n+1) \subseteq U\ n) \wedge$
$(\forall W.\ neighborhood\ \mathbf{1}\ W \longrightarrow (\exists n.\ U\ n \subseteq W))$
proof −
from *assms* **obtain** $\mathcal{B}$ **where** $h\mathcal{B}$:
$countable\ \mathcal{B} \wedge (\forall W{\in}\mathcal{B}.\ openin\ T\ W) \wedge (\forall U.\ neighborhood\ \mathbf{1}\ U \longrightarrow (\exists W{\in}\mathcal{B}.\ \mathbf{1} \in W \wedge W \subseteq U))$
unfolding *first-countable-def* **by** *fastforce*
define $\mathfrak{B} :: {}'g\ set\ set$ **where** $\mathfrak{B} = insert\ (carrier\ G)\ \{W \in \mathcal{B}.\ \mathbf{1} \in W\}$
define $B :: nat \Rightarrow {}'g\ set$ **where** $B = from\text{-}nat\text{-}into\ \mathfrak{B}$
have $\mathfrak{B} \neq \{\} \wedge (\forall W{\in}\mathfrak{B}.\ neighborhood\ \mathbf{1}\ W)$ **unfolding** $\mathfrak{B}$*-def* **using** $h\mathcal{B}$
by (*metis group-is-space insert-iff insert-not-empty mem-Collect-eq one-closed openin-topspace*)
then have *B-neighborhood*: $\forall n.\ neighborhood\ \mathbf{1}\ (B\ n)$ **unfolding** *B-def* **by** (*simp add*: *from-nat-into*)
define P **where** $P\ n\ V \longleftrightarrow V \subseteq B\ n \wedge neighborhood\ \mathbf{1}\ V \wedge symmetric\ V$ **for** $n\ V$
define Q **where** $Q\ (n :: nat)\ V\ W \longleftrightarrow W <\#> W \subseteq V$ **for** $n\ V\ W$

have ∃ *V*. *P 0 V*
proof −
obtain *W* **where** *neighborhood* **1** *W* ∧ *symmetric W* ∧ *W* <#> *W* ⊆ *B 0*
using *neighborhoods-of-1 B-neighborhood* **by** *fastforce*
moreover from *this* **have** *W* ⊆ *B 0* **using** *set-mult-one-subset open-set-in-carrier* **by** *blast*
ultimately show *?thesis* **unfolding** *P-def* **by** *auto*
qed
moreover have ∃ *W*. *P* (*Suc n*) *W* ∧ *Q n V W* **if** *P n V* **for** *n V*
proof −
have *neighborhood* **1** (*V* ∩ *B* (*Suc n*)) **using** *B-neighborhood that* **unfolding** *P-def* **by** *auto*
then obtain *W* **where** *neighborhood* **1** *W* ∧ *symmetric W* ∧ *W* <#> *W* ⊆ *V* ∩ *B* (*Suc n*)
using *neighborhoods-of-1* **by** *fastforce*
moreover from *this* **have** *W* ⊆ *B* (*Suc n*)
using *set-mult-one-subset*[*of W W*] *open-set-in-carrier*[*of W*] **by** *fast*
ultimately show *?thesis* **unfolding** *P-def Q-def* **by** *auto*
qed
ultimately obtain *U* **where** *hU*: ∀ *n*. *P n* (*U n*) ∧ *Q n* (*U n*) (*U* (*Suc n*))
using *dependent-nat-choice* **by** *metis*
moreover have ∃ *n*. *U n* ⊆ *W* **if** *neighborhood* **1** *W* **for** *W*
proof −
from *that* **obtain** *W′* **where** *hW′*: *W′* ∈ ℬ ∧ **1** ∈ *W′* ∧ *W′* ⊆ *W* **using** *hℬ* **by** *blast*
then have *W′* ∈ 𝔅 ∧ *countable* 𝔅 **unfolding** 𝔅-*def* **using** *hℬ* **by** *simp*
then obtain *n* **where** *B n* = *W′* **unfolding** *B-def* **using** *from-nat-into-to-nat-on* **by** *fast*
then show *?thesis* **using** *hW′ hU* **unfolding** *P-def* **by** *blast*
qed
ultimately show *?thesis* **unfolding** *P-def Q-def* **by** *auto*
qed

definition *left-invariant-metric* Δ ⟷ *Metric-space* (*carrier G*) Δ ∧
(∀ *σ τ ϱ*. *σ* ∈ *carrier G* ∧ *τ* ∈ *carrier G* ∧ *ϱ* ∈ *carrier G* ⟶ Δ (*ϱ* ⊗ *σ*) (*ϱ* ⊗ *τ*) = Δ *σ τ*)

definition *right-invariant-metric* Δ ⟷ *Metric-space* (*carrier G*) Δ ∧
(∀ *σ τ ϱ*. *σ* ∈ *carrier G* ∧ *τ* ∈ *carrier G* ∧ *ϱ* ∈ *carrier G* ⟶ Δ (*σ* ⊗ *ϱ*) (*τ* ⊗ *ϱ*) = Δ *σ τ*)

lemma *left-invariant-metricE*:
assumes *left-invariant-metric* Δ *σ* ∈ *carrier G τ* ∈ *carrier G ϱ* ∈ *carrier G*
shows Δ (*ϱ* ⊗ *σ*) (*ϱ* ⊗ *τ*) = Δ *σ τ*
using *assms* **unfolding** *left-invariant-metric-def* **by** *blast*

lemma *right-invariant-metricE*:
assumes *right-invariant-metric* Δ *σ* ∈ *carrier G τ* ∈ *carrier G ϱ* ∈ *carrier G*
shows Δ (*σ* ⊗ *ϱ*) (*τ* ⊗ *ϱ*) = Δ *σ τ*

using *assms* **unfolding** *right-invariant-metric-def* **by** *blast*

theorem *Birkhoff-Kakutani-left*:
assumes *Hausdorff*: *Hausdorff-space T* **and** *first-countable*: *first-countable T*
shows $\exists\,\Delta$. *left-invariant-metric* $\Delta \wedge$ *Metric-space.mtopology* (*carrier G*) $\Delta = T$
proof −
from *first-countable* **obtain** *U* :: *nat* ⇒ *'g set* **where**
U-props: $\forall\, n$. *neighborhood* **1** (*U n*) ∧ *symmetric* (*U n*) ∧ *U* (*n* + *1*) <#> *U* (*n* + *1*) ⊆ *U n* **and**
neighborhood-base: ∀ *W*. *neighborhood* **1** *W* ⟶ (∃ *n*. *U n* ⊆ *W*)
using *first-countable-neighborhoods-of-1-sequence* **by** *auto*
from *U-props* **obtain** *N* **where**
prenorm: *group-prenorm N* **and**
norm-ball-in-U: ∀ *n*. {σ ∈ *carrier G*. *N* σ < *1*/*2*^*n*} ⊆ *U n* **and**
U-in-norm-ball: ∀ *n*. *U n* ⊆ {σ ∈ *carrier G*. *N* σ ≤ *2*/*2*^*n*}
using *construction-of-prenorm-respecting-topology* **by** *meson*
have *continuous*: *continuous-map T euclideanreal N* **using** *prenorm*
proof (*rule group-prenorm-continuous-if-continuous-at-1*, *intro allI impI*)
fix ε :: *real* **assume** ε > *0*
then obtain *n* **where** *hn*: *1*/*2*^*n* < ε
by (*metis divide-less-eq-1-pos one-less-numeral-iff power-one-over real-arch-pow-inv semiring-norm*(*76*) *zero-less-numeral*)
then have *N* σ < ε **if** σ ∈ *U* (*n* + *1*) **for** σ **using** *that U-in-norm-ball* **by** *fastforce*
then show ∃ *U*. *neighborhood* **1** *U* ∧ (∀σ∈*U*. *N* σ < ε) **using** *U-props* **by** *meson*
qed
let *?B* = λε. {σ ∈ *carrier G*. *N* σ < ε}
let *?*Δ = λσ τ. *N* (*inv* σ ⊗ τ)
let *?*δ = λσ τ. *if* σ ∈ *carrier G* ∧ τ ∈ *carrier G then ?*Δ σ τ *else 42*
have *?*Δ σ τ ≥ *0* **if** σ ∈ *carrier G* ∧ τ ∈ *carrier G* **for** σ τ
using *group-prenorm-nonnegative prenorm that* **by** *blast*
moreover have *?*Δ σ τ = *?*Δ τ σ **if** σ ∈ *carrier G* ∧ τ ∈ *carrier G* **for** σ τ
proof −
have *inv* τ ⊗ σ = *inv* (*inv* σ ⊗ τ) **using** *inv-mult-group inv-inv that* **by** *auto*
then show *?thesis* **using** *prenorm that* **by** *fastforce*
qed
moreover have *?*Δ σ τ = *0* ⟷ σ = τ **if** σ ∈ *carrier G* ∧ τ ∈ *carrier G* **for** σ τ
proof
assume *?*Δ σ τ = *0*
then have *inv* σ ⊗ τ ∈ *U n* **for** *n* **using** *norm-ball-in-U that* **by** *fastforce*
then have *inv* σ ⊗ τ ∈ *W* **if** *neighborhood* **1** *W* **for** *W* **using** *neighborhood-base that* **by** *auto*
then have *inv* σ ⊗ τ = **1** **using** *Hausdorff-space-sing-Inter-opens*[*of T* **1**] *Hausdorff* **by** *blast*
then show σ = τ **using** *inv-comm inv-equality that* **by** *fastforce*
next
assume σ = τ

then show *?*Δ σ τ = *0* **using** *that prenorm* **by** *force*
qed
moreover have *?*Δ σ $\varrho \leq$ *?*Δ σ τ + *?*Δ τ ϱ **if** $\sigma \in$ *carrier G* $\wedge$ $\tau \in$ *carrier G* $\wedge$ $\varrho \in$ *carrier G* **for** σ τ ϱ
proof –
have *inv* $\sigma \otimes \varrho$ = (*inv* $\sigma \otimes \tau$) $\otimes$ (*inv* $\tau \otimes \varrho$) **using** *m-assoc*[*symmetric*] *that* **by** (*simp add*: *inv-solve-right*)
then show *?thesis* **using** *prenorm that* **by** *auto*
qed
ultimately have *metric*: *Metric-space* (*carrier G*) *?*δ **unfolding** *Metric-space-def* **by** *auto*
then interpret *Metric-space carrier G ?*δ **by** *blast*
have *?*Δ ($\varrho \otimes \sigma$) ($\varrho \otimes \tau$) = *?*Δ σ τ **if** $\sigma \in$ *carrier G* $\wedge$ $\tau \in$ *carrier G* $\wedge$ $\varrho \in$ *carrier G* **for** σ τ ϱ
proof –
have *inv* $\sigma \otimes \tau$ = *inv* ($\varrho \otimes \sigma$) $\otimes$ ($\varrho \otimes \tau$) **using** *that m-assoc*[*symmetric*] **by** (*simp add*: *inv-solve-left inv-solve-right*)
then show *?thesis* **by** *simp*
qed
then have *left-invariant*: *left-invariant-metric ?*δ
unfolding *left-invariant-metric-def* **using** *metric* **by** *auto*
have *mball-coset-of-norm-ball*: *mball* σ ε = σ <# *?B* ε **if** *h*σ: $\sigma \in$ *carrier G* **for** σ ε
proof –
have *mball* σ ε = {$\tau \in$ *carrier G*. *N* (*inv* $\sigma \otimes \tau$) < ε} **unfolding** *mball-def* **using** *h*σ **by** *auto*
also have ... = σ <# (*?B* ε)
proof –
have $\tau \in \sigma$ <# (*?B* ε) **if** $\tau \in$ *carrier G* $\wedge$ *N* (*inv* $\sigma \otimes \tau$) < ε **for** τ
proof –
have $\sigma \otimes$ (*inv* $\sigma \otimes \tau$) = τ **using** *h*σ *that* **by** (*metis inv-closed inv-solve-left m-closed*)
moreover have *inv* $\sigma \otimes \tau \in$ *?B* ε **using** *h*σ *that* **by** *fastforce*
ultimately show *?thesis* **unfolding** *l-coset-def* **by** *force*
qed
moreover have $\tau \in$ *carrier G* $\wedge$ *N* (*inv* $\sigma \otimes \tau$) < ε **if** $\tau \in \sigma$ <# (*?B* ε) **for** τ
proof –
from *that* **obtain** ϱ **where** $\varrho \in$ *?B* ε $\wedge$ $\tau = \sigma \otimes \varrho$ **unfolding** *l-coset-def* **by** *blast*
moreover from *this* **have** *inv* $\sigma \otimes \tau = \varrho$ **using** *h*σ **by** (*simp add*: *inv-solve-left'*)
ultimately show *?thesis* **using** *h*σ **by** *simp*
qed
ultimately show *?thesis* **by** *blast*
qed
finally show *?thesis* **by** *presburger*
qed
define *ball* **where** *ball S* $\longleftrightarrow$ ($\exists \sigma$ ε. $\sigma \in$ *carrier G* $\wedge$ *S* = *mball* σ ε) **for** *S*

have *openin mtopology V* **if** *ball V* **for** *V* **using** *that* **unfolding** *ball-def* **by** *fast*
moreover have ∃ *W*. *ball W* ∧ σ ∈ *W* ∧ *W* ⊆ *V* **if** *openin mtopology V* ∧ σ ∈ *V* **for** σ *V*
unfolding *ball-def* **using** *openin-mtopology that* **by** (*smt* (*verit*, *best*) *centre-in-mball-iff subset-iff*)
ultimately have *openin-metric*: *openin mtopology* = *arbitrary union-of ball*
by (*simp add*: *openin-topology-base-unique*)
have *openin T V* **if** *ball V* **for** *V*
proof −
from *that* **obtain** σ ε **where** σ ∈ *carrier G* ∧ *V* = σ <# *?B* ε
unfolding *ball-def* **using** *mball-coset-of-norm-ball* **by** *blast*
moreover have *openin T* (*?B* ε) **using** *continuous*
by (*simp add*: *continuous-map-upper-lower-semicontinuous-lt*)
ultimately show *?thesis* **using** *open-set-translations*(*1*) **by** *presburger*
qed
moreover have ∃ *W*. *ball W* ∧ σ ∈ *W* ∧ *W* ⊆ *V* **if** *neighborhood* σ *V* **for** σ *V*
proof −
from *that* **have** *in-group*: σ ∈ *carrier G* **using** *open-set-in-carrier* **by** *fast*
then have *neighborhood* **1** (*inv* σ <# *V*)
using *l-coset-def open-set-translations*(*1*) *that l-inv* **by** *fastforce*
then obtain *n* **where** *U n* ⊆ *inv* σ <# *V* **using** *neighborhood-base* **by** *presburger*
then have *?B* (*1*/*2*^*n*) ⊆ *inv* σ <# *V* **using** *norm-ball-in-U* **by** *blast*
then have σ <# *?B* (*1*/*2*^*n*) ⊆ σ <# (*inv* σ <# *V*) **unfolding** *l-coset-def* **by** *fast*
also have ... = *V* **using** *in-group that open-set-in-carrier* **by** (*simp add*: *lcos-m-assoc lcos-mult-one*)
finally have *mball* σ (*1*/*2*^*n*) ⊆ *V* **using** *mball-coset-of-norm-ball in-group* **by** *blast*
then show *?thesis* **unfolding** *ball-def*
by (*smt* (*verit*) *centre-in-mball-iff divide-pos-pos in-group one-add-one zero-less-power zero-less-two*)
qed
ultimately have *openin T* = *arbitrary union-of ball* **by** (*simp add*: *openin-topology-base-unique*)
then show *?thesis* **using** *left-invariant openin-metric topology-eq* **by** *fastforce*
qed

theorem *Birkhoff-Kakutani-right*:
assumes *Hausdorff*: *Hausdorff-space T* **and** *first-countable*: *first-countable T*
shows ∃ Δ. *right-invariant-metric* Δ ∧ *Metric-space.mtopology* (*carrier G*) Δ = *T*
proof −
from *first-countable* **obtain** *U* :: *nat* ⇒ *'g set* **where**
U-props: ∀ *n*. *neighborhood* **1** (*U n*) ∧ *symmetric* (*U n*) ∧ *U* (*n* + *1*) <#> *U* (*n* + *1*) ⊆ *U n* **and**
neighborhood-base: ∀ *W*. *neighborhood* **1** *W* ⟶ (∃ *n*. *U n* ⊆ *W*)
using *first-countable-neighborhoods-of-1-sequence* **by** *auto*
from *U-props* **obtain** *N* **where**
prenorm: *group-prenorm N* **and**

norm-ball-in-U: $\forall n.\ \{\sigma \in carrier\ G.\ N\ \sigma < 1/2\hat{}n\} \subseteq U\ n$ **and**
U-in-norm-ball: $\forall n.\ U\ n \subseteq \{\sigma \in carrier\ G.\ N\ \sigma \leq 2/2\hat{}n\}$
using *construction-of-prenorm-respecting-topology* **by** *meson*
have *continuous*: *continuous-map T euclideanreal N* **using** *prenorm*
proof (*rule group-prenorm-continuous-if-continuous-at-1*, *intro allI impI*)
fix ε :: *real* **assume** $\varepsilon > 0$
then obtain n **where** *hn*: $1/2\hat{}n < \varepsilon$
by (*metis divide-less-eq-1-pos one-less-numeral-iff power-one-over real-arch-pow-inv semiring-norm*(*76*) *zero-less-numeral*)
then have $N\ \sigma < \varepsilon$ **if** $\sigma \in U\ (n + 1)$ **for** σ **using** *that U-in-norm-ball* **by** *fastforce*
then show $\exists U.\ neighborhood\ \mathbf{1}\ U \wedge (\forall \sigma \in U.\ N\ \sigma < \varepsilon)$ **using** *U-props* **by** *meson*
qed
let $?B = \lambda\varepsilon.\ \{\sigma \in carrier\ G.\ N\ \sigma < \varepsilon\}$
let $?\Delta = \lambda\sigma\ \tau.\ N\ (\sigma \otimes inv\ \tau)$
let $?\delta = \lambda\sigma\ \tau.\ if\ \sigma \in carrier\ G \wedge \tau \in carrier\ G\ then\ ?\Delta\ \sigma\ \tau\ else\ 42$
have $?\Delta\ \sigma\ \tau \geq 0$ **if** $\sigma \in carrier\ G \wedge \tau \in carrier\ G$ **for** $\sigma\ \tau$
using *group-prenorm-nonnegative prenorm that* **by** *blast*
moreover have $?\Delta\ \sigma\ \tau = ?\Delta\ \tau\ \sigma$ **if** $\sigma \in carrier\ G \wedge \tau \in carrier\ G$ **for** $\sigma\ \tau$
proof −
have $\tau \otimes inv\ \sigma = inv\ (\sigma \otimes inv\ \tau)$ **using** *inv-mult-group inv-inv that* **by** *auto*
then show *?thesis* **using** *prenorm that* **by** *auto*
qed
moreover have $?\Delta\ \sigma\ \tau = 0 \longleftrightarrow \sigma = \tau$ **if** $\sigma \in carrier\ G \wedge \tau \in carrier\ G$ **for** $\sigma\ \tau$
proof
assume $?\Delta\ \sigma\ \tau = 0$
then have $\sigma \otimes inv\ \tau \in U\ n$ **for** n **using** *norm-ball-in-U that* **by** *fastforce*
then have $\sigma \otimes inv\ \tau \in W$ **if** *neighborhood* $\mathbf{1}$ W **for** W **using** *neighborhood-base that* **by** *auto*
then have $\sigma \otimes inv\ \tau = \mathbf{1}$ **using** *Hausdorff-space-sing-Inter-opens*[*of T* $\mathbf{1}$] *Hausdorff* **by** *blast*
then show $\sigma = \tau$ **using** *inv-equality that* **by** *fastforce*
next
assume $\sigma = \tau$
then show $?\Delta\ \sigma\ \tau = 0$ **using** *that prenorm* **by** *force*
qed
moreover have $?\Delta\ \sigma\ \varrho \leq ?\Delta\ \sigma\ \tau + ?\Delta\ \tau\ \varrho$ **if** $\sigma \in carrier\ G \wedge \tau \in carrier\ G \wedge \varrho \in carrier\ G$ **for** $\sigma\ \tau\ \varrho$
proof −
have $\sigma \otimes inv\ \varrho = (\sigma \otimes inv\ \tau) \otimes (\tau \otimes inv\ \varrho)$ **using** *m-assoc that* **by** (*simp add*: *inv-solve-left*)
then show *?thesis* **using** *prenorm that* **by** *auto*
qed
ultimately have *metric*: *Metric-space* (*carrier G*) $?\delta$ **unfolding** *Metric-space-def* **by** *auto*
then interpret *Metric-space carrier G* $?\delta$ **by** *blast*
have $?\Delta\ (\sigma \otimes \varrho)\ (\tau \otimes \varrho) = ?\Delta\ \sigma\ \tau$ **if** $\sigma \in carrier\ G \wedge \tau \in carrier\ G \wedge \varrho \in$

carrier G **for** σ τ ϱ
proof −
have σ ⊗ inv τ = (σ ⊗ ϱ) ⊗ inv (τ ⊗ ϱ) **using** that m-assoc **by** (simp add: inv-solve-left inv-solve-right)
then show ?thesis **by** simp
qed
then have right-invariant: right-invariant-metric ?δ
unfolding right-invariant-metric-def **using** metric **by** auto
have mball-coset-of-norm-ball: mball σ ε = ?B ε #> σ **if** hσ: σ ∈ carrier G **for** σ ε
proof −
have mball σ ε = {τ ∈ carrier G. N (σ ⊗ inv τ) < ε} **unfolding** mball-def **using** hσ **by** auto
also have ... = (?B ε) #> σ
proof −
have τ ∈ (?B ε) #> σ **if** τ ∈ carrier G ∧ N (σ ⊗ inv τ) < ε **for** τ
proof −
have inv (σ ⊗ inv τ) ⊗ σ = τ **using** hσ that **by** (simp add: inv-mult-group m-assoc)
moreover have inv (σ ⊗ inv τ) ∈ ?B ε **using** hσ that prenorm **by** fastforce
ultimately show ?thesis **unfolding** r-coset-def **by** force
qed
moreover have τ ∈ carrier G ∧ N (σ ⊗ inv τ) < ε **if** τ ∈ (?B ε) #> σ **for** τ
proof −
from that **obtain** ϱ **where** ϱ ∈ ?B ε ∧ τ = ϱ ⊗ σ **unfolding** r-coset-def **by** blast
moreover from this **have** σ ⊗ inv τ = inv ϱ **using** hσ
by (metis (no-types, lifting) inv-closed inv-mult-group inv-solve-left m-closed mem-Collect-eq)
ultimately show ?thesis **using** hσ prenorm **by** fastforce
qed
ultimately show ?thesis **by** blast
qed
finally show ?thesis **by** presburger
qed
define ball **where** ball S ⟷ (∃σ ε. σ ∈ carrier G ∧ S = mball σ ε) **for** S
have openin mtopology V **if** ball V **for** V **using** that **unfolding** ball-def **by** fast
moreover have ∃ W. ball W ∧ σ ∈ W ∧ W ⊆ V **if** openin mtopology V ∧ σ ∈ V **for** σ V
unfolding ball-def **using** openin-mtopology that **by** (smt (verit, best) centre-in-mball-iff subset-iff)
ultimately have openin-metric: openin mtopology = arbitrary union-of ball
by (simp add: openin-topology-base-unique)
have openin T V **if** ball V **for** V
proof −
from that **obtain** σ ε **where** σ ∈ carrier G ∧ V = ?B ε #> σ
unfolding ball-def **using** mball-coset-of-norm-ball **by** blast
moreover have openin T (?B ε) **using** continuous

by (*simp add*: *continuous-map-upper-lower-semicontinuous-lt*)
ultimately show *?thesis* **using** *open-set-translations*(*2*) **by** *presburger*
qed
moreover have ∃ *W*. *ball W* ∧ σ ∈ *W* ∧ *W* ⊆ *V* **if** *neighborhood* σ *V* **for** σ *V*
proof −
from *that* **have** *in-group*: σ ∈ *carrier G* **using** *open-set-in-carrier* **by** *fast*
then have *neighborhood* **1** (*V* #> *inv* σ)
using *r-coset-def open-set-translations*(*2*) *that r-inv* **by** *fastforce*
then obtain *n* **where** *U n* ⊆ *V* #> *inv* σ **using** *neighborhood-base* **by** *presburger*
then have *?B* (*1*/*2*^*n*) ⊆ *V* #> *inv* σ **using** *norm-ball-in-U* **by** *blast*
then have *?B* (*1*/*2*^*n*) #> σ ⊆ (*V* #> *inv* σ) #> σ **unfolding** *r-coset-def* **by** *fast*
also have ... = *V* **using** *in-group that open-set-in-carrier* **by** (*simp add*: *coset-mult-assoc*)
finally have *mball* σ (*1*/*2*^*n*) ⊆ *V* **using** *mball-coset-of-norm-ball in-group* **by** *blast*
then show *?thesis* **unfolding** *ball-def*
by (*smt* (*verit*) *centre-in-mball-iff divide-pos-pos in-group one-add-one zero-less-power zero-less-two*)
qed
ultimately have *openin T* = *arbitrary union-of ball* **by** (*simp add*: *openin-topology-base-unique*)
then show *?thesis* **using** *right-invariant openin-metric topology-eq* **by** *fastforce*
qed

corollary *Birkhoff-Kakutani-iff*:
shows *metrizable-space T* ⟷ *Hausdorff-space T* ∧ *first-countable T*
using *Birkhoff-Kakutani-left Metric-space.metrizable-space-mtopology metrizable-imp-Hausdorff-space*
metrizable-imp-first-countable **unfolding** *left-invariant-metric-def* **by** *metis*

end

end

3 Examples of Topological Groups

theory *Topological-Group-Examples*
imports *Topological-Group*
begin

Summary This section gives examples of topological groups.

lemma (**in** *group*) *discrete-topological-group*:
shows *topological-group G* (*discrete-topology* (*carrier G*))
proof −
let *?T* = *discrete-topology* (*carrier G*)
have *topspace ?T* = *carrier G* **using** *topspace-discrete-topology* **by** *force*
moreover have *continuous-map* (*prod-topology ?T ?T*) *?T* (λ(σ,τ). σ ⊗ τ)

unfolding *prod-topology-discrete-topology*[*symmetric*] **by** *auto*
ultimately show *?thesis* **unfolding** *topological-group-def topological-group-axioms-def*
by *fastforce*
qed

lemma *topological-group-real-power-space*:
defines $\mathfrak{R}$ :: (*real*^′*n*) *monoid* ≡ (|*carrier* = *UNIV*, *mult* = (+), *one* = *0*|)
defines *T* :: (*real*^′*n*) *topology* ≡ *euclidean*
shows *topological-group* $\mathfrak{R}$ *T*
proof −
have $x \in$ *Units* $\mathfrak{R}$ **for** *x*
proof −
have $x \otimes_{\mathfrak{R}} -x = \mathbf{1}_{\mathfrak{R}}$ $-x \otimes_{\mathfrak{R}} x = \mathbf{1}_{\mathfrak{R}}$ **using** $\mathfrak{R}$-*def* **by** *auto*
then show *?thesis* **unfolding** *Units-def* $\mathfrak{R}$-*def* **by** *fastforce*
qed
then have *group*: *group* $\mathfrak{R}$ **by** (*unfold-locales*) (*auto simp*: $\mathfrak{R}$-*def*)
then interpret *group* $\mathfrak{R}$ **by** *auto*
have *group-is-space*: *topspace T* = *carrier* $\mathfrak{R}$
unfolding $\mathfrak{R}$-*def T-def* **by** *force*
have *mul-continuous*: *continuous-map* (*prod-topology T T*) *T* (λ(*x*,*y*). $x \otimes_{\mathfrak{R}} y$)
using *continuous-map-add*[*OF continuous-map-fst continuous-map-snd*]
unfolding *T-def* $\mathfrak{R}$-*def* **by** (*simp add*: *case-prod-beta*′)
have $(-x) \otimes_{\mathfrak{R}} x = \mathbf{1}_{\mathfrak{R}}$ **for** *x* **unfolding** $\mathfrak{R}$-*def* **by** *auto*
then have $inv_{\mathfrak{R}}\ x = -x$ **for** *x* **using** *inv-equality* $\mathfrak{R}$-*def* **by** *simp*
moreover have *continuous-map T T uminus* **unfolding** *T-def* **by** *force*
ultimately have *continuous-map T T* ($\lambda x.\ inv_{\mathfrak{R}}\ x$) **by** *simp*
then show *?thesis* **using** *group-is-space mul-continuous group*
unfolding *topological-group-def topological-group-axioms-def* **by** *blast*
qed

definition *unit-group* :: (′*a* :: *field*) *monoid* **where**
unit-group = (|*carrier* = *UNIV* − {*0*}, *mult* = (∗), *one* = *1*|)

lemma
group-unit-group: *group unit-group* **and**
inv-unit-group: $x \in$ *carrier unit-group* ⟹ $inv_{unit\text{-}group}\ x$ = *inverse x*
proof −
have $x \in$ *Units unit-group* **if** $x \neq 0$ **for** *x*
proof −
have $x \otimes_{unit\text{-}group} 1/x = \mathbf{1}_{unit\text{-}group}$ $1/x \otimes_{unit\text{-}group} x = \mathbf{1}_{unit\text{-}group}$
using *that* **unfolding** *unit-group-def* **by** *auto*
then show *?thesis* **unfolding** *Units-def unit-group-def* **using** *that* **by** *fastforce*
qed
then show *group unit-group* **by** (*unfold-locales*) (*auto simp*: *unit-group-def*)
then interpret *group unit-group* **by** *blast*
show $inv_{unit\text{-}group}\ x$ = *inverse x* **if** $x \in$ *carrier unit-group*
using *that inv-equality*[*of inverse x*] **unfolding** *unit-group-def* **by** *simp*
qed

```
lemma topological-group-real-unit-group:
  defines T :: real topology ≡ subtopology euclidean (UNIV − {0})
  shows topological-group unit-group T
proof −
  let ?ℜ = unit-group :: real monoid
   have group-is-space: topspace T = carrier ?ℜ unfolding unit-group-def T-def
by force
   have continuous-map (prod-topology euclidean euclidean) euclidean (λ(x,y). x
⊗?ℜ y)
    using continuous-map-real-mult[OF continuous-map-fst continuous-map-snd]
    unfolding T-def unit-group-def by (simp add: case-prod-beta′)
  then have continuous-map (prod-topology T T) euclideanreal (λ(x,y). x ⊗?ℜ y)
   unfolding T-def subtopology-Times[symmetric] using continuous-map-from-subtopology
by blast
  moreover have (λ(x,y). x ⊗?ℜ y) ∈ topspace (prod-topology T T) → UNIV −
{0}
    unfolding T-def unit-group-def by fastforce
 ultimately have mul-continuous: continuous-map (prod-topology T T) T (λ(x,y).
x ⊗?ℜ y)
    unfolding T-def using continuous-map-into-subtopology by blast
  have continuous-map T euclideanreal inverse
    using continuous-map-real-inverse[of T id] unfolding T-def by auto
  moreover have inverse ∈ topspace T → topspace T unfolding T-def by fast-
force
  ultimately have continuous-map T T inverse
    unfolding T-def using continuous-map-into-subtopology by auto
  then have continuous-map T T (λx. inv?ℜ x)
    using group-is-space continuous-map-eq inv-unit-group by metis
  then show ?thesis using group-is-space mul-continuous group-unit-group
    unfolding topological-group-def topological-group-axioms-def by blast
qed

end
```

4 Matrix groups

```
theory Matrix-Group
  imports
    Topological-Group
    Topological-Group-Examples
    HOL−Analysis.Determinants
begin
```

Summary In this section we define the general linear group and some of its subgroups. We also introduce topologies on vector types and use them to prove the aforementioned groups to be topological groups.

4.1 Topologies on vector types

definition *vec-topology* :: *'a topology* ⇒ (*'a^'n*) *topology* **where**
vec-topology T = *quot-topology* (*product-topology* (λ*i. T*) *UNIV*) *vec-lambda*

lemma *producttop-vectop-homeo*:
 shows *homeomorphic-map* (*product-topology* (λ*i. T*) *UNIV*) (*vec-topology T*) *vec-lambda*
proof −
 have *inj-on vec-lambda* (*topspace* (*product-topology* (λ*i. T*) *UNIV*)) **unfolding** *inj-on-def* **by** *force*
 then show *?thesis* **unfolding** *vec-topology-def*
  using *injective-quotient-map-homeo*[*OF projection-quotient-map*] **by** *blast*
qed

lemma *homeo-inverse-homeo*:
 assumes *homeo*: *homeomorphic-map X Y f* **and** *fg-id*: ∀ *y* ∈ *topspace Y*. *f* (*g y*) = *y* **and**
  g-image: ∀ *y* ∈ *topspace Y*. *g y* ∈ *topspace X*
 shows *homeomorphic-map Y X g*
proof −
 from *homeo* **obtain** *h* **where**
  h-homeo: *homeomorphic-map Y X h* **and** *hf-id*: (∀ *x* ∈ *topspace X*. *h* (*f x*) = *x*)
  by (*smt* (*verit*) *homeomorphic-map-maps homeomorphic-maps-map*)
 have *g y* = *h y* **if** *y* ∈ *topspace Y* **for** *y*
 proof −
  have *g y* = *h* (*f* (*g y*)) **using** *hf-id that g-image* **by** *fastforce*
  then show *?thesis* **using** *fg-id that* **by** *simp*
 qed
 then show *?thesis* **using** *homeomorphic-map-eq*[*OF h-homeo*] **by** *presburger*
qed

lemma *vectop-producttop-homeo*:
 shows *homeomorphic-map* (*vec-topology T*) (*product-topology* (λ*i. T*) *UNIV*) *vec-nth*
proof −
 let *?T'* = *product-topology* (λ*i. T*) *UNIV*
 have *vec-lambda* (*vec-nth v*) = *v* **for** *v* :: *'a^'n* **by** *simp*
 moreover have *vec-nth v* ∈ *topspace ?T'* **if** *v* ∈ *topspace* (*vec-topology T*) **for** *v* :: *'a^'n*
 proof −
  have ∃ *f* ∈ *topspace ?T'*. *v* = *vec-lambda f* **using** *that*
   unfolding *vec-topology-def topspace-quot-topology image-def* **by** *fast*
  then show *?thesis* **by** *fastforce*
 qed
 ultimately show *?thesis* **using** *homeo-inverse-homeo*[*OF producttop-vectop-homeo*] **by** *blast*
qed

lemma *vec-topology-euclidean* [*simp*]:

defines T :: ($'a$ :: *topological-space*) *topology* ≡ *euclidean*
defines T_{vec} :: ($'a$^$'n$) *topology* ≡ *euclidean*
shows *vec-topology* $T = T_{vec}$
proof −
have *openin* (*vec-topology* T) U **if** *openin* T_{vec} U **for** U
proof −
have *hU*: *open* U **using** *open-openin that* **unfolding** T_{vec}-*def* **by** *blast*
have $\exists U'$. *openin* (*vec-topology* T) $U' \wedge x \in U' \wedge U' \subseteq U$ **if** $x \in U$ **for** x
proof −
from *that hU* **obtain** V :: $'n \Rightarrow 'a$ *set* **where**
hV: $(\forall i.$ *open* $(V\ i) \wedge x\$i \in V\ i) \wedge (\forall y. (\forall i. y\$i \in V\ i) \longrightarrow y \in U)$
unfolding *open-vec-def* **by** *force*
let *?W* = Π_E $i\in$*UNIV*. $V\ i$
from *hV* **have** *openin* T $(V\ i)$ **for** i **using** *open-openin* **unfolding** *T-def* **by** *blast*
then have *openin* (*product-topology* ($\lambda i.$ T) *UNIV*) *?W* **by** (*simp add*: *openin-PiE*)
then have *is-open*: *openin* (*vec-topology* T) (*vec-lambda*‘*?W*)
using *producttop-vectop-homeo homeomorphic-map-openness openin-subset* **by** *metis*
have *vec-nth* $x \in$ *?W* **using** *hV* **by** *fast*
then have *contains-x*: $x \in$ (*vec-lambda*‘*?W*) **unfolding** *image-def* **by** *force*
have $y \in U$ **if** *vec-nth* $y \in$ *?W* **for** y
proof −
from *that* **have** $y\$i \in V\ i$ **for** i **by** *fast*
then show *?thesis* **using** *hV* **by** *blast*
qed
then have (*vec-lambda*‘*?W*) $\subseteq U$ **by** *force*
then show *?thesis* **using** *contains-x is-open* **by** *meson*
qed
then show *?thesis* **by** (*meson openin-subopen*)
qed
moreover have *openin* T_{vec} U **if** *openin* (*vec-topology* T) U **for** U
proof −
from *that* **have** *hU*: *openin* (*product-topology* ($\lambda i.$ T) *UNIV*) (*vec-nth*‘U)
using *vectop-producttop-homeo homeomorphic-map-openness openin-subset* **by** *metis*
have $\exists V. (\forall i.$ *open* $(V\ i) \wedge x\ \$\ i \in V\ i) \wedge (\forall y. (\forall i. y\ \$\ i \in V\ i) \longrightarrow y \in U)$ **if** $x \in U$ **for** x
proof −
from *that* **have** *vec-nth* $x \in$ (*vec-nth*‘U) **unfolding** *image-def* **by** *blast*
then obtain V :: $'n \Rightarrow 'a$ *set*
where *hV*: $(\forall i.$ *openin* T $(V\ i)) \wedge$ *vec-nth* $x \in (\Pi_E$ $i\in$*UNIV*. $V\ i) \wedge (\Pi_E$ $i\in$*UNIV*. $V\ i) \subseteq$ (*vec-nth*‘U)
using *hU product-topology-open-contains-basis* **by** (*metis* (*no-types*, *lifting*))
then have *open* $(V\ i) \wedge x\$i \in V\ i$ **for** i **unfolding** *T-def* **using** *open-openin* **by** *fast*
moreover have $y \in U$ **if** $\forall i. y\$i \in V\ i$ **for** y
proof −

have *vec-nth y* ∈ (Π_E *i*∈*UNIV*. *V i*) **using** *that* **by** *blast*
then show *?thesis* **using** *hV* **by** (*metis image-iff in-mono vec-nth-inject*)
qed
ultimately show *?thesis* **by** *blast*
qed
then have *open U* **unfolding** *open-vec-def* **by** *blast*
then show *?thesis* **unfolding** T_{vec}*-def* **using** *open-openin* **by** *blast*
qed
ultimately show *?thesis* **using** *topology-eq* **by** *meson*
qed

lemma *vec-projection-continuous*:
shows *continuous-map* (*vec-topology T*) *T* (λ*v*. *v*$*i*)
using *homeomorphic-imp-continuous-map*[*OF vectop-producttop-homeo*] **by** *fast*

lemma *vec-components-continuous-imp-continuous*:
fixes *f* :: *'x* ⇒ *'a*^*'n*
assumes ∀ *i*. *continuous-map X T* (λ*x*. (*f x*) $ *i*)
shows *continuous-map X* (*vec-topology T*) *f*
proof −
have *continuous-map X* (*product-topology* (λ*i*. *T*) *UNIV*) (*vec-nth* ∘ *f*) **using** *assms* **by** *auto*
moreover have *f* = *vec-lambda* ∘ (*vec-nth* ∘ *f*) **by** *fastforce*
ultimately show *?thesis* **using** *continuous-map-compose*
homeomorphic-imp-continuous-map[*OF producttop-vectop-homeo*] **by** *fastforce*
qed

definition *matrix-topology* :: *'a topology* ⇒ (*'a*^*'n*^*'m*) *topology* **where**
matrix-topology T = *vec-topology* (*vec-topology T*)

lemma *matrix-topology-euclidean*[*simp*]:
shows *matrix-topology euclidean* = *euclidean*
unfolding *matrix-topology-def* **by** *simp*

lemma *matrix-projection-continuous*:
shows *continuous-map* (*matrix-topology T*) *T* (λ*A*. *A*$*i*$*j*)
proof −
have (λ*A*. *A*$*i*$*j*) = (λ*x*. *x*$*j*) ∘ (λ*A*. *A*$*i*) **by** *fastforce*
then show *?thesis* **unfolding** *matrix-topology-def*
using *vec-projection-continuous continuous-map-compose* **by** *metis*
qed

lemma *matrix-components-continuous-imp-continuous*:
fixes *f* :: *'x* ⇒ *'a*^*'n*^*'m*
assumes ⋀*i j*. *continuous-map X T* (λ*x*. (*f x*) $ *i* $ *j*)
shows *continuous-map X* (*matrix-topology T*) *f*
unfolding *matrix-topology-def* **using** *vec-components-continuous-imp-continuous* *assms* **by** *metis*

4.2 The general linear group as a topological group

definition *GL* :: ((′*a* :: *field*)^′*n*^′*n*) *monoid* **where**
GL = ⦇*carrier* = {*A*. *invertible A*}, *monoid.mult* = (∗∗), *one* = *mat 1*⦈

definition *GL-topology* :: (*real*^′*n*^′*n*) *topology* **where**
GL-topology = *subtopology euclidean* (*carrier GL*)

lemma *topspace-GL*: *topspace GL-topology* = {*A*. *invertible A*}
  unfolding *GL-topology-def topspace-subtopology GL-def* **by** *simp*

4.2.1 Continuity of matrix operations

lemma *det-continuous*:
  defines *T* :: (*real*^′*n*^′*n*) *topology* ≡ *euclidean*
  shows *continuous-map T euclideanreal det*
proof −
  let *?T′* = *matrix-topology euclideanreal*
  let *?S* = {π. π *permutes* (*UNIV* :: ′*n set*)}
  have *S-finite*: *finite ?S* **by** *simp*
  have *finite* (*UNIV* :: ′*n set*) **by** *simp*
  then have *continuous-map ?T′ euclideanreal* (λ*A*. ∏ *i* ∈ (*UNIV* :: ′*n set*). (*A* $ *i* $ π *i*))
    for π :: ′*n* ⇒ ′*n* **using** *continuous-map-prod*[*OF - matrix-projection-continuous*] **by** *fast*
   then have *continuous-map ?T′ euclideanreal* (λ*A*. *of-int* (*sign* π) ∗ (∏ *i* ∈ (*UNIV* :: ′*n set*). (*A* $ *i* $ π *i*)))
    for π :: ′*n* ⇒ ′*n* **using** *continuous-map-real-mult-left* **by** *fast*
  from *continuous-map-sum*[*OF S-finite this*] **have** *continuous-map ?T′ euclideanreal*
    (λ*A*. ∑ π∈*?S*. *of-int* (*sign* π) ∗ (∏ *i* ∈ (*UNIV* :: ′*n set*). *A* $ *i* $ π *i*)) **by** *fast*
  then show *?thesis* **unfolding** *T-def matrix-topology-euclidean det-def* **by** *force*
qed

lemma *matrix-mul-continuous*:
  defines *T1* :: (*real*^′*n*^′*m*) *topology* ≡ *euclidean*
  defines *T2* :: (*real*^′*r*^′*n*) *topology* ≡ *euclidean*
  defines *T3* :: (*real*^′*r*^′*m*) *topology* ≡ *euclidean*
  shows *continuous-map* (*prod-topology T1 T2*) *T3* (λ(*A*,*B*). *A* ∗∗ *B*)
proof −
  let *?T* = *prod-topology T1 T2*
  have *continuous-map ?T euclideanreal* (λ*AB*. (*fst AB* ∗∗ *snd AB*) $ *i* $ *j*) **for** *i* :: ′*m* **and** *j* :: ′*r*
  proof −
    have *eq*: (λ*AB*. (*fst AB* ∗∗ *snd AB*) $ *i* $ *j*) = (λ*AB*. (∑(*k*::′*n*)∈*UNIV*. *fst AB* $ *i* $ *k* ∗ *snd AB* $ *k* $ *j*))
      unfolding *matrix-matrix-mult-def* **by** *auto*
    have
      comp1: (λ*AB*. *fst AB* $ *i* $ *k*) = (λ*A*. *A*$*i*$*k*) ∘ *fst* **and**
      comp2: (λ*AB*. *snd AB* $ *k* $ *j*) = (λ*B*. *B*$*k*$*j*) ∘ *snd*

for *k* :: *'n* **by** *auto*
from *comp1* **have** *continuous-map ?T euclideanreal* (*λAB. fst AB* $ *i* $ *k*) **for** *k* :: *'n*
unfolding *T1-def matrix-topology-euclidean*[*symmetric*]
using *continuous-map-compose*[*OF continuous-map-fst matrix-projection-continuous*] **by** *metis*
moreover from *comp2* **have** *continuous-map ?T euclideanreal* (*λAB. snd AB* $ *k* $ *j*) **for** *k* :: *'n*
unfolding *T2-def matrix-topology-euclidean*[*symmetric*]
using *continuous-map-compose*[*OF continuous-map-snd matrix-projection-continuous*] **by** *metis*
ultimately have *summand-continuous*:
continuous-map ?T euclideanreal (*λAB. fst AB* $ *i* $ *k* ∗ *snd AB* $ *k* $ *j*) **for** *k* :: *'n*
using *continuous-map-real-mult* **by** *blast*
have *finite*: *finite* (*UNIV* :: *'n set*) **by** *simp*
have *continuous-map ?T euclideanreal* (*λAB.* (∑ (*k*::*'n*)∈*UNIV. fst AB* $ *i* $ *k* ∗ *snd AB* $ *k* $ *j*))
using *continuous-map-sum*[*OF finite summand-continuous*] **by** *fast*
then show *?thesis* **unfolding** *eq* **by** *blast*
qed
from *matrix-components-continuous-imp-continuous*[*OF this*] **show** *?thesis*
unfolding *T3-def matrix-topology-euclidean*[*symmetric*] **by** (*simp add*: *case-prod-beta'*)
qed

lemma *transpose-continuous*:
shows *continuous-map* (*euclidean* :: ((*'a* :: *topological-space*)^*'n*^*'m*) *topology*) *euclidean transpose*
proof −
have *continuous-map euclidean euclidean* (*λA.* (*transpose A*) $ *i* $ *j*) **for** *i* :: *'n* **and** *j* :: *'m*
unfolding *transpose-def matrix-topology-euclidean*[*symmetric*]
using *matrix-projection-continuous*[*of euclidean j i*] **by** *fastforce*
from *matrix-components-continuous-imp-continuous*[*OF this*] **show** *?thesis*
unfolding *matrix-topology-euclidean* **by** *blast*
qed

4.2.2 Continuity of matrix inversion

lemma *matrix-mul-columns*:
fixes *A* :: (*'a* :: *semiring-1*)^*'n*^*'m* **and** *B* :: *'a*^*'k*^*'n*
shows *column j* (*A* ∗∗ *B*) = *A* ∗*v* (*column j B*)
unfolding *column-def matrix-matrix-mult-def matrix-vector-mult-def* **by** *force*

lemma *matrix-columns-unique*:
assumes ∀ *j. column j A* = *column j B*
shows *A* = *B*
using *assms* **unfolding** *column-def* **by** (*simp add*: *vec-eq-iff*)

lemma *matrix-inv-is-inv*:
  assumes *invertible A*
  shows *A* ∗∗ (*matrix-inv A*) = *mat 1* **and** (*matrix-inv A*) ∗∗ *A* = *mat 1*
proof −
  show *A* ∗∗ *matrix-inv A* = *mat 1*
   using *assms* **unfolding** *invertible-def matrix-inv-def* **by** (*simp add*: *verit-sko-ex′*)
  show (*matrix-inv A*) ∗∗ *A* = *mat 1*
   using *assms* **unfolding** *invertible-def matrix-inv-def* **by** (*simp add*: *verit-sko-ex′*)
qed

lemma *invertible-imp-right-inverse-is-inverse*:
  assumes *invertible*: *invertible A* **and** *A* ∗∗ *B* = *mat 1*
  shows *matrix-inv A* = *B*
   using *matrix-inv-is-inv*[*OF invertible*] *assms* **by** (*metis matrix-mul-assoc matrix-mul-lid*)

lemma *matrix-inv-invertible*:
  assumes *invertible A*
  shows *invertible* (*matrix-inv A*)
  using *assms matrix-inv-is-inv invertible-def* **by** *fast*

lemma *det-inv*:
  fixes *A* :: (′*a* :: *field*)^′*n*^′*n*
  assumes *det A* ≠ *0*
  shows *det* (*matrix-inv A*) = *1* / *det A*
proof −
 have *A* ∗∗ (*matrix-inv A*) = *mat 1* **using** *assms invertible-det-nz matrix-inv-is-inv*(*1*) **by** *fast*
  then have *det A* ∗ *det* (*matrix-inv A*) = *1* **using** *det-mul*[*of A matrix-inv A*] **by** *auto*
  then show *?thesis* **using** *assms* **by** (*metis nonzero-mult-div-cancel-left*)
qed

See proposition "cramer" from HOL-Analysis.Determinants

definition *cramer-inv* :: (′*a* :: *field*)^′*n*^′*n* ⇒ ′*a*^′*n*^′*n* **where**
cramer-inv A = (χ *i j*. *det*(χ *k l*. *if l* = *i then* (*axis j 1*) $ *k else A*$*k*$*l*) / *det A*)

lemma *cramer-inv-is-inverse*:
  assumes *invertible*: *invertible* (*A* :: (′*a* :: *field*)^′*n*^′*n*)
  shows *matrix-inv A* = *cramer-inv A*
proof −
  have *A* ∗∗ (*cramer-inv A*) = *mat 1*
  proof −
   have *column j* (*cramer-inv A*) = (χ *i*. *det*(χ *k l*. *if l* = *i then* (*axis j 1*) $ *k else A*$*k*$*l*) / *det A*) **for** *j*
     unfolding *cramer-inv-def column-def* **by** *simp*
    moreover have *det A* ≠ *0* **using** *invertible* **unfolding** *invertible-det-nz* **by** *force*
   ultimately have *A* ∗*v* (*column j* (*cramer-inv A*)) = *axis j 1* **for** *j* **using** *cramer*

by *auto*
then have *column j* (*A* ∗∗ (*cramer-inv A*)) = *axis j 1* **for** *j* **unfolding** *matrix-mul-columns* **by** *auto*
moreover have *column j* (*mat 1*) = *axis j 1* **for** *j* :: *'n* **unfolding** *column-def mat-def axis-def* **by** *simp*
ultimately show *?thesis* **using** *matrix-columns-unique* **by** *metis*
qed
then show *?thesis* **using** *invertible invertible-imp-right-inverse-is-inverse* **unfolding** *GL-def* **by** *fastforce*
qed

lemma *matrix-inv-continuous*:
shows *continuous-map* (*GL-topology* :: (*real^'n^'n*) *topology*) *GL-topology matrix-inv*
proof −
define *B* :: *real^'n^'n* ⇒ *'n* ⇒ *'n* ⇒ *'n* ⇒ *'n* ⇒ *real* **where**
B = (*λA i j k l. if l = i then* (*axis j 1*) $ *k else A*$*k*$*l*)
define *C* :: *real^'n^'n* ⇒ *'n* ⇒ *'n* ⇒ *real^'n^'n* **where**
C A i j = (*χ k l. B A i j k l*) **for** *A i j*
have *det-GL-continuous*: *continuous-map GL-topology euclideanreal det*
unfolding *GL-topology-def* **using** *continuous-map-from-subtopology*[*OF det-continuous*] **by** *fast*
have *continuous-map euclidean euclideanreal* (*λA. B A i j k l*) **for** *i j k l*
proof (*cases l = i*)
case *True*
then have (*λA. B A i j k l*) = (*λA.* (*axis j 1*) $ *k*) **unfolding** *B-def* **by** *force*
moreover have *continuous-map euclidean euclideanreal* (*λA.* (*axis j 1*) $ *k*) **by** *simp*
ultimately show *?thesis* **by** (*smt* (*verit*) *continuous-map-eq*)
next
case *False*
then have (*λA. B A i j k l*) = (*λA. A*$*k*$*l*) **unfolding** *B-def* **by** *simp*
then show *?thesis* **unfolding** *matrix-topology-euclidean*[*symmetric*]
using *matrix-projection-continuous*[*of euclideanreal k l*] **by** *force*
qed
then have *continuous-map euclidean euclideanreal* (*λA.* (*C A i j*) $ *k* $ *l*)
for *i j k l* **unfolding** *C-def* **by** *simp*
from *matrix-components-continuous-imp-continuous*[*OF this*]
have *continuous-map euclidean euclidean* (*λA. C A i j*) **for** *i j*
unfolding *matrix-topology-euclidean*[*symmetric*] **by** *blast*
from *continuous-map-compose*[*OF this det-continuous*]
have *continuous-map euclidean euclideanreal* (*λA. det* (*C A i j*)) **for** *i j* **by** *force*
then have *continuous-map GL-topology euclideanreal* (*λA. det* (*C A i j*)) **for** *i j*
unfolding *GL-topology-def* **using** *continuous-map-from-subtopology* **by** *fast*
from *continuous-map-real-divide*[*OF this det-GL-continuous*]
have *continuous-map GL-topology euclideanreal* (*λA. det* (*C A i j*) / *det A*) **for** *i j*
unfolding *topspace-GL invertible-det-nz* **by** *simp*
then have *continuous-map GL-topology euclideanreal* (*λA.* (*χ i j. det* (*C A i j*)

/ *det A*) $ *i* $ *j*) **for** *i j* **by** *simp*
from *matrix-components-continuous-imp-continuous*[*OF this*]
have *continuous-map* (*GL-topology* :: (*real*^'*n*^'*n*) *topology*) *euclidean cramer-inv*

unfolding *cramer-inv-def C-def B-def matrix-topology-euclidean*[*symmetric*] **by** *blast*
from *continuous-map-eq*[*OF this*] **have** *continuous-map* (*GL-topology* :: (*real*^'*n*^'*n*) *topology*) *euclidean matrix-inv*
unfolding *topspace-GL* **using** *cramer-inv-is-inverse* **by** (*metis mem-Collect-eq*)
moreover have *matrix-inv A* ∈ *topspace GL-topology* **if** *A* ∈ *topspace GL-topology* **for** *A* :: *real*^'*n*^'*n*
using *that* **unfolding** *topspace-GL*
by (*metis invertible-imp-right-inverse-is-inverse invertible-left-inverse invertible-right-inverse mem-Collect-eq*)
ultimately show *?thesis* **unfolding** *GL-topology-def Pi-def image-def* **using** *continuous-map-into-subtopology* **by** *auto*
qed

4.2.3 The general linear group is topological

lemma
GL-group: *group GL* **and**
GL-carrier [*simp*]: *carrier GL* = {*A*. *invertible A*} **and**
GL-inv [*simp*]: *A* ∈ *carrier GL* ⟹ inv_{GL} *A* = *matrix-inv A*
proof −
show *carrier GL* = {*A*. *invertible A*} **unfolding** *GL-def* **by** *simp*
show *group GL*
proof (*unfold-locales*, *goal-cases*)
case *3*
then show *?case* **unfolding** *GL-def* **by** (*simp add*: *invertible-def*)
case *6*
then show *?case* **using** *GL-def* **unfolding** *Units-def invertible-def*
by (*smt* (*verit*, *ccfv-threshold*) *Collect-mono invertible-def mem-Collect-eq monoid.select-convs*(*1*) *monoid.select-convs*(*2*) *partial-object.select-convs*(*1*))
qed (*unfold GL-def*, *auto simp*: *matrix-mul-assoc invertible-mult*)
interpret *group GL* **by** *fact*
show *A* ∈ *carrier GL* ⟹ inv_{GL} *A* = *matrix-inv A*
using *matrix-inv-is-inv matrix-inv-invertible inv-equality* **unfolding** *GL-def* **by** *fastforce*
qed

lemma
GL-topological-group: *topological-group GL GL-topology* **and**
GL-open: *openin* (*euclidean* :: (*real*^'*n*^'*n*) *topology*) (*carrier GL*)
proof −
have *group-is-space*: *topspace GL-topology* = *carrier GL* **unfolding** *topspace-GL GL-def* **by** *simp*
have *continuous-map* (*prod-topology GL-topology GL-topology*) *euclidean* (λ(*A*,*B*). *A* ∗∗ *B*)

unfolding *GL-topology-def subtopology-Times*[*symmetric*] **using** *matrix-mul-continuous continuous-map-from-subtopology* **by** *fast*
from *continuous-map-into-subtopology*[*OF this*]
have *continuous-map* (*prod-topology GL-topology GL-topology*) *GL-topology* (λ(*A*,*B*). *A* $\otimes_{GL}$ *B*)
unfolding *GL-topology-def Pi-def topspace-prod-topology topspace-subtopology GL-def* **using** *invertible-mult* **by** *auto*
moreover from *continuous-map-eq*[*OF matrix-inv-continuous*]
have *continuous-map GL-topology GL-topology* (λ*A*. *inv*$_{GL}$ *A*) **unfolding** *group-is-space* **using** *GL-inv* **by** *metis*
ultimately show *topological-group GL GL-topology* **using** *GL-group group-is-space*
unfolding *topological-group-def topological-group-axioms-def* **by** *blast*
have *openin euclideanreal* ((*topspace euclideanreal*) $-$ {*0*}) **by** *auto*
from *openin-continuous-map-preimage*[*OF det-continuous this*]
have *openin euclidean* {(*A* :: *real*^'*n*^'*n*) $\in$ *topspace euclidean*. *det A* $\in$ ((*topspace euclideanreal*) $-$ {*0*})} **by** *blast*
moreover have *carrier GL* = {*A* :: *real*^'*n*^'*n*. *det A* $\neq$ *0*}
using *group-is-space*[*symmetric*] *invertible-det-nz* **unfolding** *topspace-GL* **by** *blast*
ultimately show *openin* (*euclidean* :: (*real*^'*n*^'*n*) *topology*) (*carrier GL*) **by** *fastforce*
qed

4.3 Subgroups of the general linear group

definition *SL* :: ((*'a* :: *field*)^'*n*^'*n*) *monoid* **where**
SL = *GL* (|*carrier* := {*A*. *det A* = *1*}|)

lemma *det-homomorphism*: *group-hom GL unit-group det*
proof $-$
have *det* $\in$ *carrier GL* $\to$ *carrier unit-group*
unfolding *GL-carrier unit-group-def* **using** *invertible-det-nz* **by** *fastforce*
moreover have *det* (*A* $\otimes_{GL}$ *B*) = *det A* $\otimes_{unit\text{-}group}$ *det B* **for** *A B*
unfolding *GL-def unit-group-def* **using** *det-mul* **by** *auto*
ultimately have *det* $\in$ *hom GL unit-group* **unfolding** *hom-def* **by** *blast*
then show *?thesis* **using** *GL-group group-unit-group*
unfolding *group-hom-def group-hom-axioms-def* **by** *blast*
qed

lemma
SL-kernel-det: *carrier* (*SL* :: ((*'a* :: *field*)^'*n*^'*n*) *monoid*) = *kernel GL unit-group det* **and**
SL-subgroup: *subgroup* (*carrier SL*) (*GL* :: (*'a*^'*n*^'*n*) *monoid*) **and**
SL-carrier [*simp*]: *carrier SL* = {*A*. *det A* = *1*}
proof $-$
interpret *group-hom GL* :: (*'a*^'*n*^'*n*) *monoid unit-group det* **using** *det-homomorphism* **by** *blast*
show *carrier SL* = {*A*. *det A* = *1*} **unfolding** *SL-def* **by** *simp*
then show *carrier* (*SL* :: (*'a*^'*n*^'*n*) *monoid*) = *kernel GL unit-group det*

unfolding *kernel-def GL-carrier unit-group-def* **using** *invertible-det-nz* **by** *force*
then show *subgroup* (*carrier SL*) (*GL* :: ('*a*^'*n*^'*n*) *monoid*) **using** *subgroup-kernel*
by *presburger*
qed

lemma
SL-topological-group: *topological-group SL* (*subtopology GL-topology* (*carrier SL*))
and
SL-closed: *closedin GL-topology* (*carrier SL*)
proof −
interpret *topological-group GL GL-topology* **using** *GL-topological-group* **by** *blast*
show *topological-group SL* (*subtopology GL-topology* (*carrier SL*))
unfolding *SL-def* **using** *topological-subgroup*[*OF SL-subgroup*] **by** *force*
have *closedin euclideanreal* {*1*} **by** *simp*
then have *closedin GL-topology* {*A* ∈ *topspace GL-topology*. *det A* = *1*} **unfolding** *GL-topology-def*
using *continuous-map-from-subtopology*[*OF det-continuous*] *closedin-continuous-map-preimage*

by (*smt* (*verit*, *ccfv-SIG*) *Collect-cong singleton-iff*)
moreover have {*A* ∈ *topspace GL-topology*. *det A* = *1*} = {*A*. *det A* = *1*}
using *topspace-GL* **using** *invertible-det-nz* **by** *fastforce*
ultimately show *closedin GL-topology* (*carrier SL*) **unfolding** *SL-carrier* **by** (*smt* (*verit*))
qed

definition *GO* :: (*real*^'*n*^'*n*) *monoid* **where**
GO = *GL* (|*carrier* := {*A*. *orthogonal-matrix A*}|)

lemma
GO-subgroup: *subgroup* {*A* :: *real*^'*n*^'*n*. *orthogonal-matrix A*} *GL* **and**
GO-carrier [*simp*]: *carrier GO* = {*A*. *orthogonal-matrix A*}
proof −
show *carrier GO* = {*A*. *orthogonal-matrix A*} **unfolding** *GO-def* **by** *force*
show *subgroup* {*A* :: *real*^'*n*^'*n*. *orthogonal-matrix A*} *GL*
proof (*unfold-locales*, *goal-cases*)
case *1*
then show *?case* **unfolding** *GL-carrier orthogonal-matrix-def invertible-def* **by** *blast*
next
case (*2 A B*)
then show *?case* **unfolding** *GL-def* **using** *orthogonal-matrix-mul*[*of A B*] **by** *force*
next
case *3*
then show *?case* **unfolding** *GL-def* **using** *orthogonal-matrix-id* **by** *simp*
next
case (*4 A*)
then have *A* ∈ *carrier GL* **unfolding** *GL-carrier orthogonal-matrix-def invertible-def* **by** *blast*

moreover from *4* **have** *orthogonal-matrix* (*matrix-inv A*)
by (*metis invertible-imp-right-inverse-is-inverse invertible-right-inverse mem-Collect-eq orthogonal-matrix-def orthogonal-matrix-transpose*)
ultimately show *?case* **using** *GL-inv* **by** *fastforce*
qed
qed

lemma
GO-topological-group: *topological-group GO* (*subtopology GL-topology* (*carrier GO*))
and
GO-closed: *closedin* (*GL-topology* :: (*real*^'*n*^'*n*) *topology*) (*carrier GO*)
proof −
interpret *topological-group GL GL-topology* **using** *GL-topological-group* **by** *blast*
show *topological-group GO* (*subtopology GL-topology* (*carrier GO*))
unfolding *GO-def* **using** *topological-subgroup*[*OF GO-subgroup*] **by** *simp*
have *one-closed*: *closedin euclidean* {(*mat 1*) :: *real*^'*n*^'*n*} **by** *fastforce*
have *continuous-map euclidean* (*prod-topology euclidean euclidean*) (λ*A* :: *real*^'*n*^'*n*. (*transpose A*, *A*))
using *continuous-map-pairedI*[*OF transpose-continuous continuous-map-id*] **by** *force*
from *continuous-map-compose*[*OF this matrix-mul-continuous*]
have *continuous-map euclidean euclidean* (λ*A* :: *real*^'*n*^'*n*. (*transpose A*) ∗∗ *A*) **by** *force*
from *closedin-continuous-map-preimage*[*OF this one-closed*]
have *closedin euclidean* {*A* :: *real*^'*n*^'*n*. (*transpose A*) ∗∗ *A* = *mat 1*} **by** *force*
moreover have *carrier GO* = {*A* :: *real*^'*n*^'*n*. (*transpose A*) ∗∗ *A* = *mat 1*}
using *orthogonal-matrix* **unfolding** *GO-carrier* **by** *blast*
ultimately have *closedin* (*euclidean* :: (*real*^'*n*^'*n*) *topology*) (*carrier GO*) **by** (*smt* (*verit*, *del-insts*))
moreover have *carrier GO* ⊆ *carrier GL*
unfolding *GO-carrier GL-carrier orthogonal-matrix-def invertible-def* **by** *blast*
ultimately show *closedin* (*GL-topology* :: (*real*^'*n*^'*n*) *topology*) (*carrier GO*)
unfolding *GL-topology-def* **using** *closedin-subset-topspace* **by** *blast*
qed

definition *SO* :: (*real*^'*n*^'*n*) *monoid* **where**
SO = *GL* (|*carrier* := {*A*. *orthogonal-matrix A* ∧ *det A* = *1*}|)

lemma
SO-carrier [*simp*]: *carrier SO* = {*A*. *orthogonal-matrix A* ∧ *det A* = *1*} **and**
SO-subgroup: *subgroup* {*A* :: *real*^'*n*^'*n*. *orthogonal-matrix A* ∧ *det A* = *1*} *GL*
proof −
show *carrier SO* = {*A*. *orthogonal-matrix A* ∧ *det A* = *1*} **unfolding** *SO-def* **by** *auto*
have *eq*: {*A* :: *real*^'*n*^'*n*. *orthogonal-matrix A* ∧ *det A* = *1*} = {*A*. *orthogonal-matrix A*} ∩ {*A*. *det A* = *1*} **by** *fastforce*
show *subgroup* {*A* :: *real*^'*n*^'*n*. *orthogonal-matrix A* ∧ *det A* = *1*} *GL*
unfolding *eq* **using** *subgroup-intersection*[*OF GO-subgroup SL-subgroup*] **by** *simp*

qed

lemma
SO-topological-group: *topological-group SO* (*subtopology GL-topology* (*carrier SO*))
and
SO-closed: *closedin GL-topology* (*carrier SO*)
proof−
interpret *topological-group GL GL-topology* **using** *GL-topological-group* **by** *blast*
show *topological-group SO* (*subtopology GL-topology* (*carrier SO*))
unfolding *SO-def* **using** *topological-subgroup*[*OF SO-subgroup*] **by** *simp*
have *carrier SO* = *carrier SL* ∩ *carrier GO* **unfolding** *SO-carrier SL-carrier GO-carrier* **by** *blast*
then show *closedin GL-topology* (*carrier SO*) **using** *closedin-Int*[*OF SL-closed GO-closed*] **by** *metis*
qed

end

References

[1] A. Arhangelskii and M. Tkachenko. *Topological Groups and Related Structures*. Atlantis Studies in Mathematics 1. Atlantis Press, 2008.

[2] D. Bump. *Lie Groups*. Graduate Texts in Mathematics 225. Springer, 2 edition, 2013.

[3] R. (https://math.stackexchange.com/users/464495/randall). Why is the quotient group a topological group? Mathematics Stack Exchange. Accessed on 2024-07-23.

[4] I. James. *Topological and uniform spaces*. Undergraduate texts in mathematics. Springer, 1 edition, 1987.

[5] J. Prem. Topologische gruppen und haarsches mass. Universität Regensburg, 4 2013. Accessed on 2024-07-22.