The Fundamental Group of the Circle

Arthur Freitas Ramos David Barros Hulak
Ruy J. G. B. de Queiroz

September 2, 2026

Abstract

We formalise the classical theorem of algebraic topology that the fundamental group of the circle is isomorphic to the additive group of the integers, $\pi_1(S^1) \cong \mathbb{Z}$. The circle is modelled as the unit sphere in the complex plane with basepoint 1, and the carrier of the fundamental group is the set of path-homotopy classes of loops based at 1, with concatenation as the group operation. The group laws are obtained from the homotopy groupoid laws of the Isabelle analysis library. The isomorphism with $\mathbb{Z}$ is given by the degree map, sending a homotopy class to the winding number about the origin of any representative loop; that this is a bijective group homomorphism follows from the winding-number classification of loops in the punctured plane, together with a radial retraction onto the circle. The entry is self-contained on top of the Isabelle distribution.

AI assistance was used for proof engineering. The final definitions, statements, and proofs are checked by Isabelle.

Contents

1 Introduction

The fundamental group of the circle is one of the first non-trivial computations in algebraic topology: it is the infinite cyclic group, so $\pi_1(S^1) \cong \mathbb{Z}$. The result underlies many classical theorems, including the Brouwer fixed-point theorem in dimension two and the fundamental theorem of algebra. Standard textbook treatments can be found in Hatcher [1] and Munkres [2].

Rather than developing covering-space theory, this entry follows an analytic route that is directly supported by the Isabelle distribution. The circle is modelled as the unit sphere in the complex plane with basepoint 1, and the group is built from path-homotopy classes of based loops. The isomorphism with $\mathbb{Z}$ is realised by the *degree* map, which sends a homotopy class to the winding number about the origin of any representative loop. The winding-number theory of the Isabelle analysis library provides exactly the facts needed: winding numbers are integer valued on loops, additive under concatenation, and classify loops in the punctured plane up to homotopy. A radial retraction of the punctured plane onto the circle transfers homotopies back to the circle, which yields injectivity of the degree map.

2 The Fundamental Group of the Circle

theory *Fundamental-Group-Circle*
  imports
    HOL−Complex-Analysis.Complex-Analysis
    HOL−Algebra.Group
begin

We formalise the classical result of algebraic topology that the fundamental group of the circle is isomorphic to the additive group of integers, $\pi_1(S^1) \cong \mathbb{Z}$.

The development is self-contained on top of the Isabelle distribution. It models the circle as the unit sphere in the complex plane, with basepoint *1*. Based loops are paths in the circle starting and ending at *1*, and the carrier of the fundamental group is the set of path-homotopy classes of such loops, with concatenation as the group operation. The group structure is obtained directly from the homotopy "groupoid" laws in *HOL−Analysis.Homotopy.*

The isomorphism with $\mathbb{Z}$ is realised by the *degree* map, which sends a homotopy class to the winding number of any representative loop about the origin. That this is a well-defined bijective homomorphism rests on the classification of loops in the punctured plane up to homotopy by their winding number, from *HOL−Complex-Analysis.Riemann-Mapping*, together with a radial retraction bridging homotopies in the punctured plane and in the circle.

AI assistance was used for proof engineering. The final definitions, statements, and proofs are checked by Isabelle.

2.1 The circle and based loops

definition *circle* :: *complex set* **where**
circle = *sphere 0 1*

lemma *one-in-circle* [*simp*]: (*1*::*complex*) ∈ *circle*
by (*simp add*: *circle-def dist-norm*)

lemma *zero-notin-circle* [*simp*]: (*0*::*complex*) ∉ *circle*
by (*simp add*: *circle-def*)

lemma *circle-subset-punctured*: *circle* ⊆ − {*0*}
by (*auto simp*: *circle-def*)

lemma *norm-of-circle*: *z* ∈ *circle* ⟹ *norm z* = *1*
by (*simp add*: *circle-def dist-norm*)

A (based) loop is a path in the circle starting and finishing at *1*.

definition *is-loop* :: (*real* ⇒ *complex*) ⇒ *bool* **where**
is-loop p ⟷ *path p* ∧ *path-image p* ⊆ *circle* ∧ *pathstart p* = *1* ∧ *pathfinish p* = *1*

lemma *is-loop-path*: *is-loop p* ⟹ *path p*
by (*simp add*: *is-loop-def*)

lemma *is-loop-image*: *is-loop p* ⟹ *path-image p* ⊆ *circle*
by (*simp add*: *is-loop-def*)

lemma *is-loop-pathstart*: *is-loop p* ⟹ *pathstart p* = *1*
by (*simp add*: *is-loop-def*)

lemma *is-loop-pathfinish*: *is-loop p* ⟹ *pathfinish p* = *1*
by (*simp add*: *is-loop-def*)

lemma *is-loop-selfloop*: *is-loop p* ⟹ *pathfinish p* = *pathstart p*
by (*simp add*: *is-loop-def*)

lemma *is-loop-connect*: ⟦*is-loop p*; *is-loop q*⟧ ⟹ *pathfinish p* = *pathstart q*
by (*simp add*: *is-loop-def*)

lemma *is-loop-zero-notin-image*: *is-loop p* ⟹ *0* ∉ *path-image p*
using *is-loop-image zero-notin-circle* **by** *blast*

lemma *is-loop-linepath-1* [*simp*]: *is-loop* (*linepath 1 1*)
by (*auto simp*: *is-loop-def path-image-linepath*)

lemma *is-loop-join* [*simp*]:
assumes *is-loop p* **and** *is-loop q*
shows *is-loop* (*p* +++ *q*)
proof −

from *assms*(*1*) **have** *p*: *path p path-image p* $\subseteq$ *circle pathstart p* = *1 pathfinish p* = *1*
  by (*simp-all add*: *is-loop-def*)
from *assms*(*2*) **have** *q*: *path q path-image q* $\subseteq$ *circle pathstart q* = *1 pathfinish q* = *1*
  by (*simp-all add*: *is-loop-def*)
have *fs*: *pathfinish p* = *pathstart q* **using** *p*(*4*) *q*(*3*) **by** *simp*
have *path* (*p* +++ *q*) **using** *p*(*1*) *q*(*1*) *fs* **by** (*rule path-join-imp*)
moreover have *path-image* (*p* +++ *q*) $\subseteq$ *circle*
  using *fs p*(*2*) *q*(*2*) **by** (*simp add*: *path-image-join*)
moreover have *pathstart* (*p* +++ *q*) = *1* **using** *p*(*3*) **by** *simp*
moreover have *pathfinish* (*p* +++ *q*) = *1* **using** *q*(*4*) **by** *simp*
ultimately show *?thesis* **by** (*simp add*: *is-loop-def*)
qed

lemma *is-loop-reversepath* [*simp*]:
assumes *is-loop p*
shows *is-loop* (*reversepath p*)
using *assms* **by** (*simp add*: *is-loop-def*)

2.2 Homotopy classes of loops

definition *loop-class* :: (*real* ⇒ *complex*) ⇒ (*real* ⇒ *complex*) *set* **where**
loop-class p = {*q*. *is-loop q* ∧ *homotopic-paths circle p q*}

definition *loops-carrier* :: (*real* ⇒ *complex*) *set set* **where**
loops-carrier = {*loop-class p* |*p*. *is-loop p*}

definition *rep* :: (*real* ⇒ *complex*) *set* ⇒ (*real* ⇒ *complex*) **where**
rep A = (*SOME p*. *is-loop p* ∧ *A* = *loop-class p*)

lemma *in-loop-class*: *is-loop p* ⟹ *p* ∈ *loop-class p*
by (*simp add*: *loop-class-def homotopic-paths-refl is-loop-path is-loop-image*)

lemma *loop-class-eqI*:
assumes *is-loop p is-loop q homotopic-paths circle p q*
shows *loop-class p* = *loop-class q*
proof −
have *homotopic-paths circle p r* = *homotopic-paths circle q r* **for** *r*
  using *assms*(*3*)
  by (*meson homotopic-paths-sym homotopic-paths-trans*)
then show *?thesis*
  by (*auto simp*: *loop-class-def*)
qed

lemma *loop-class-eq-iff*:
assumes *is-loop p is-loop q*
shows *loop-class p* = *loop-class q* ⟷ *homotopic-paths circle p q*
proof

assume *loop-class p = loop-class q*
then have *q ∈ loop-class p*
using *in-loop-class[OF assms(2)]* **by** *simp*
then show *homotopic-paths circle p q*
by (*simp add*: *loop-class-def*)
next
assume *homotopic-paths circle p q*
with *assms* **show** *loop-class p = loop-class q*
by (*rule loop-class-eqI*)
qed

lemma *loop-class-in-carrier* [*simp*]: *is-loop p* ⟹ *loop-class p* ∈ *loops-carrier*
by (*auto simp*: *loops-carrier-def*)

lemma *rep-props*:
assumes *A* ∈ *loops-carrier*
shows *is-loop* (*rep A*) ∧ *loop-class* (*rep A*) = *A*
proof −
from *assms* **obtain** *p* **where** *is-loop p* ∧ *A* = *loop-class p*
by (*auto simp*: *loops-carrier-def*)
then have ∃ *p*. *is-loop p* ∧ *A* = *loop-class p* **by** *blast*
then have *is-loop* (*rep A*) ∧ *A* = *loop-class* (*rep A*)
unfolding *rep-def* **by** (*rule someI-ex*)
then show *?thesis* **by** *auto*
qed

lemma *is-loop-rep*: *A* ∈ *loops-carrier* ⟹ *is-loop* (*rep A*)
using *rep-props* **by** *blast*

lemma *loop-class-rep*: *A* ∈ *loops-carrier* ⟹ *loop-class* (*rep A*) = *A*
using *rep-props* **by** *blast*

2.3 The fundamental group of the circle

definition *pi1-circle* :: (*real* ⇒ *complex*) *set monoid* **where**
pi1-circle =
(| *carrier* = *loops-carrier*,
mult = (λ*A B*. *loop-class* (*rep A* +++ *rep B*)),
one = *loop-class* (*linepath 1 1*) |)

lemma *carrier-pi1* [*simp*]: *carrier pi1-circle* = *loops-carrier*
by (*simp add*: *pi1-circle-def*)

lemma *one-pi1* [*simp*]: $\mathbf{1}_{pi1\text{-}circle}$ = *loop-class* (*linepath 1 1*)
by (*simp add*: *pi1-circle-def*)

Multiplication of classes is concatenation, independent of representatives.

lemma *pi1-mult-class*:
assumes *p*: *is-loop p* **and** *q*: *is-loop q*

shows *loop-class p* $\otimes_{pi1\text{-}circle}$ *loop-class q* = *loop-class* (*p* +++ *q*)
proof −
let *?rp* = *rep* (*loop-class p*) **and** *?rq* = *rep* (*loop-class q*)
have *lp*: *is-loop ?rp* **and** *cp*: *loop-class ?rp* = *loop-class p*
using *rep-props*[*OF loop-class-in-carrier*[*OF p*]] **by** *auto*
have *lq*: *is-loop ?rq* **and** *cq*: *loop-class ?rq* = *loop-class q*
using *rep-props*[*OF loop-class-in-carrier*[*OF q*]] **by** *auto*
have *hp*: *homotopic-paths circle ?rp p*
using *loop-class-eq-iff*[*OF lp p*] *cp* **by** *simp*
have *hq*: *homotopic-paths circle ?rq q*
using *loop-class-eq-iff*[*OF lq q*] *cq* **by** *simp*
have *homotopic-paths circle* (*?rp* +++ *?rq*) (*p* +++ *q*)
using *hp hq is-loop-connect*[*OF lp lq*] **by** (*intro homotopic-paths-join*) *simp-all*
then have *loop-class* (*?rp* +++ *?rq*) = *loop-class* (*p* +++ *q*)
by (*rule loop-class-eqI*[*OF is-loop-join*[*OF lp lq*] *is-loop-join*[*OF p q*]])
then show *?thesis*
by (*simp add*: *pi1-circle-def*)
qed

lemma *group-pi1-circle*: *group pi1-circle*
proof (*rule groupI*)
fix *x y* **assume** *x* ∈ *carrier pi1-circle y* ∈ *carrier pi1-circle*
then obtain *p q* **where** *is-loop p x* = *loop-class p is-loop q y* = *loop-class q*
by (*auto simp*: *loops-carrier-def*)
then show *x* $\otimes_{pi1\text{-}circle}$ *y* ∈ *carrier pi1-circle*
by (*simp add*: *pi1-mult-class*)
next
show $\mathbf{1}_{pi1\text{-}circle}$ ∈ *carrier pi1-circle*
by *simp*
next
fix *x y z*
assume *x* ∈ *carrier pi1-circle y* ∈ *carrier pi1-circle z* ∈ *carrier pi1-circle*
then obtain *p q r*
where *L*: *is-loop p is-loop q is-loop r*
and *X*: *x* = *loop-class p y* = *loop-class q z* = *loop-class r*
by (*auto simp*: *loops-carrier-def*)
have *lhs*: (*x* $\otimes_{pi1\text{-}circle}$ *y*) $\otimes_{pi1\text{-}circle}$ *z* = *loop-class* ((*p* +++ *q*) +++ *r*)
using *L X* **by** (*simp add*: *pi1-mult-class*)
have *rhs*: *x* $\otimes_{pi1\text{-}circle}$ (*y* $\otimes_{pi1\text{-}circle}$ *z*) = *loop-class* (*p* +++ (*q* +++ *r*))
using *L X* **by** (*simp add*: *pi1-mult-class*)
have *homotopic-paths circle* (*p* +++ (*q* +++ *r*)) ((*p* +++ *q*) +++ *r*)
using *L is-loop-connect*[*OF L*(*1*) *L*(*2*)] *is-loop-connect*[*OF L*(*2*) *L*(*3*)]
by (*intro homotopic-paths-assoc*) (*simp-all add*: *is-loop-path is-loop-image*)
then have *loop-class* (*p* +++ (*q* +++ *r*)) = *loop-class* ((*p* +++ *q*) +++ *r*)
by (*rule loop-class-eqI*[*OF is-loop-join*[*OF L*(*1*) *is-loop-join*[*OF L*(*2*) *L*(*3*)]]
is-loop-join[*OF is-loop-join*[*OF L*(*1*) *L*(*2*)] *L*(*3*)]]])
with *lhs rhs*
show (*x* $\otimes_{pi1\text{-}circle}$ *y*) $\otimes_{pi1\text{-}circle}$ *z* = *x* $\otimes_{pi1\text{-}circle}$ (*y* $\otimes_{pi1\text{-}circle}$ *z*)
by *simp*

next
  fix *x* **assume** $x \in$ *carrier pi1-circle*
  then obtain *p* **where** *p*: *is-loop p* **and** *X*: $x =$ *loop-class p*
    by (*auto simp*: *loops-carrier-def*)
  have *homotopic-paths circle* (*linepath 1 1* +++ *p*) *p*
    using *p is-loop-pathstart*[*OF p*]
    by (*intro homotopic-paths-lid′*) (*simp-all add*: *is-loop-path is-loop-image*)
  then have *loop-class* (*linepath 1 1* +++ *p*) = *loop-class p*
    by (*rule loop-class-eqI*[*OF is-loop-join*[*OF is-loop-linepath-1 p*] *p*])
  then show $\mathbf{1}_{\textit{pi1-circle}} \otimes_{\textit{pi1-circle}} x = x$
    using *p X* **by** (*simp add*: *pi1-mult-class*)
next
  fix *x* **assume** $x \in$ *carrier pi1-circle*
  then obtain *p* **where** *p*: *is-loop p* **and** *X*: $x =$ *loop-class p*
    by (*auto simp*: *loops-carrier-def*)
  have *homotopic-paths circle* (*reversepath p* +++ *p*) (*linepath 1 1*)
    using *p homotopic-paths-linv*[*of p circle*] *is-loop-pathfinish*[*OF p*]
    by (*simp add*: *is-loop-path is-loop-image*)
  then have *loop-class* (*reversepath p* +++ *p*) = *loop-class* (*linepath 1 1*)
   by (*rule loop-class-eqI*[*OF is-loop-join*[*OF is-loop-reversepath*[*OF p*] *p*] *is-loop-linepath-1*])
  then have *loop-class* (*reversepath p*) $\otimes_{\textit{pi1-circle}} x = \mathbf{1}_{\textit{pi1-circle}}$
    using *p X* **by** (*simp add*: *pi1-mult-class*)
  moreover have *loop-class* (*reversepath p*) $\in$ *carrier pi1-circle*
    using *p* **by** *simp*
  ultimately
  show $\exists\, y \in$ *carrier pi1-circle*. $y \otimes_{\textit{pi1-circle}} x = \mathbf{1}_{\textit{pi1-circle}}$
    by *blast*
qed

2.4 The integers as a group

definition *integer-group* :: *int monoid* **where**
  integer-group = (| *carrier* = *UNIV*, *mult* = (+), *one* = *0* |)

lemma *carrier-integer-group* [*simp*]: *carrier integer-group* = *UNIV*
  by (*simp add*: *integer-group-def*)

lemma *mult-integer-group* [*simp*]: $x \otimes_{\textit{integer-group}} y = x + y$
  by (*simp add*: *integer-group-def*)

lemma *one-integer-group* [*simp*]: $\mathbf{1}_{\textit{integer-group}} = 0$
  by (*simp add*: *integer-group-def*)

lemma *group-integer-group*: *group integer-group*
proof (*rule groupI*)
  fix *x* **assume** $x \in$ *carrier integer-group*
  show $\exists\, y \in$ *carrier integer-group*. $y \otimes_{\textit{integer-group}} x = \mathbf{1}_{\textit{integer-group}}$
    by (*rule bexI*[*of - − x*]) *simp-all*
qed *simp-all*

2.5 The winding-number degree

definition *wnd* :: (*real* ⇒ *complex*) ⇒ *int* **where**
wnd p = (*SOME n. winding-number p 0* = *of-int n*)

lemma *wnd-eq*:
assumes *is-loop p*
shows *winding-number p 0* = *of-int* (*wnd p*)
proof −
have *winding-number p 0* ∈ ℤ
using *is-loop-path*[*OF assms*] *is-loop-selfloop*[*OF assms*] *is-loop-zero-notin-image*[*OF assms*]
by (*intro integer-winding-number*) *simp-all*
then have ∃ *n. winding-number p 0* = *of-int n*
by (*auto simp*: *Ints-def*)
then show *?thesis*
unfolding *wnd-def* **by** (*rule someI-ex*)
qed

lemma *wnd-join*:
assumes *is-loop p is-loop q*
shows *wnd* (*p* +++ *q*) = *wnd p* + *wnd q*
proof −
have *winding-number* (*p* +++ *q*) *0* = *winding-number p 0* + *winding-number q 0*
using *is-loop-path*[*OF assms*(*1*)] *is-loop-path*[*OF assms*(*2*)]
is-loop-zero-notin-image[*OF assms*(*1*)] *is-loop-zero-notin-image*[*OF assms*(*2*)]
is-loop-connect[*OF assms*(*1*) *assms*(*2*)]
by (*intro winding-number-join*) *simp-all*
also have ... = *of-int* (*wnd p* + *wnd q*)
using *assms* **by** (*simp add*: *wnd-eq*)
finally have *winding-number* (*p* +++ *q*) *0* = *of-int* (*wnd p* + *wnd q*) **.**
moreover have *winding-number* (*p* +++ *q*) *0* = *of-int* (*wnd* (*p* +++ *q*))
using *assms* **by** (*simp add*: *wnd-eq*)
ultimately show *?thesis*
by (*metis of-int-eq-iff*)
qed

lemma *wnd-reversepath*:
assumes *is-loop p*
shows *wnd* (*reversepath p*) = − *wnd p*
proof −
have *winding-number* (*reversepath p*) *0* = − *winding-number p 0*
using *assms is-loop-zero-notin-image*
by (*simp add*: *winding-number-reversepath is-loop-path*)
also have ... = *of-int* (− *wnd p*)
using *assms* **by** (*simp add*: *wnd-eq*)
finally have *winding-number* (*reversepath p*) *0* = *of-int* (− *wnd p*) **.**
moreover have *winding-number* (*reversepath p*) *0* = *of-int* (*wnd* (*reversepath p*))

```
    using assms by (simp add: wnd-eq)
  ultimately show ?thesis
    by (metis of-int-eq-iff)
qed

lemma wnd-linepath-1: wnd (linepath 1 1) = 0
proof -
  have winding-number (linepath 1 1) (0::complex) = 0
    by (simp add: winding-number-trivial)
  moreover have winding-number (linepath 1 1) (0::complex) = of-int (wnd
(linepath 1 1))
    by (simp add: wnd-eq)
  ultimately show ?thesis by simp
qed

lemma wnd-homotopic:
  assumes is-loop p is-loop q homotopic-paths circle p q
  shows wnd p = wnd q
proof -
  have homotopic-paths (- {0}) p q
    using assms(3) circle-subset-punctured by (rule homotopic-paths-subset)
  moreover have
    (winding-number p 0 = winding-number q 0) = homotopic-paths (- {0}) p q
    using is-loop-path[OF assms(1)] is-loop-path[OF assms(2)]
      is-loop-zero-notin-image[OF assms(1)] is-loop-zero-notin-image[OF assms(2)]
      is-loop-pathstart[OF assms(1)] is-loop-pathstart[OF assms(2)]
      is-loop-pathfinish[OF assms(1)] is-loop-pathfinish[OF assms(2)]
    by (intro winding-number-homotopic-paths-eq) simp-all
  ultimately have winding-number p 0 = winding-number q 0 by simp
  then have of-int (wnd p) = of-int (wnd q)
    using assms by (simp add: wnd-eq)
  then show ?thesis by (metis of-int-eq-iff)
qed

definition deg :: (real ⇒ complex) set ⇒ int where
  deg A = wnd (rep A)

lemma deg-class:
  assumes is-loop p
  shows deg (loop-class p) = wnd p
proof -
  have A: loop-class p ∈ loops-carrier
    using assms by simp
  have lr: is-loop (rep (loop-class p))
    and cr: loop-class (rep (loop-class p)) = loop-class p
    using rep-props[OF A] by auto
  have hom: homotopic-paths circle (rep (loop-class p)) p
    using loop-class-eq-iff[OF lr assms] cr by simp
  have wnd (rep (loop-class p)) = wnd p
```

```
    by (rule wnd-homotopic[OF lr assms hom])
  then show ?thesis
    by (simp add: deg-def)
qed
```

2.6 The degree is a homomorphism

```
lemma deg-mult:
  assumes A ∈ loops-carrier B ∈ loops-carrier
  shows deg (A ⊗pi1-circle B) = deg A + deg B
proof −
  from assms obtain p q
    where p: is-loop p A = loop-class p and q: is-loop q B = loop-class q
    by (auto simp: loops-carrier-def)
  have deg (A ⊗pi1-circle B) = deg (loop-class (p +++ q))
    using p q by (simp add: pi1-mult-class)
  also have . . . = wnd (p +++ q)
    using p q by (simp add: deg-class)
  also have . . . = wnd p + wnd q
    using p q by (simp add: wnd-join)
  also have . . . = deg A + deg B
    using p q by (simp add: deg-class)
  finally show ?thesis .
qed

lemma deg-hom: deg ∈ hom pi1-circle integer-group
  by (rule homI) (simp-all add: deg-mult)
```

2.7 The degree is injective

A radial retraction of the punctured plane onto the circle. It fixes the circle pointwise and lets us transfer a homotopy in the punctured plane back into the circle.

```
definition retr :: complex ⇒ complex where
  retr z = z / of-real (norm z)

lemma retr-fixes: z ∈ circle =⇒ retr z = z
  by (simp add: retr-def norm-of-circle)

lemma continuous-on-retr: continuous-on (− {0}) retr
  unfolding retr-def by (intro continuous-intros) auto

lemma retr-into-circle: retr ∈ (− {0}) → circle
proof
  fix z :: complex assume z ∈ − {0}
  then have z ≠ 0 by simp
  then have norm (retr z) = 1
    by (simp add: retr-def norm-divide)
  then show retr z ∈ circle
```

```
    by (simp add: circle-def dist-norm)
qed

lemma homotopic-paths-retr-self:
  assumes is-loop p
  shows homotopic-paths circle p (retr ∘ p)
proof (rule homotopic-paths-eq)
  show path p using assms by (rule is-loop-path)
  show path-image p ⊆ circle using assms by (rule is-loop-image)
  fix t :: real assume t ∈ {0..1}
  then have p t ∈ path-image p
    by (auto simp: path-image-def)
  then have p t ∈ circle using assms is-loop-image by blast
  then show p t = (retr ∘ p) t
    by (simp add: retr-fixes)
qed

lemma deg-inj: inj-on deg (carrier pi1-circle)
proof (rule inj-onI)
  fix A B
  assume A ∈ carrier pi1-circle B ∈ carrier pi1-circle and eq: deg A = deg B
  then obtain p q
    where p: is-loop p A = loop-class p and q: is-loop q B = loop-class q
    by (auto simp: loops-carrier-def)
  from eq p q have wnd p = wnd q
    by (simp add: deg-class)
  then have winding-number p 0 = winding-number q 0
    using p q by (simp add: wnd-eq)
  moreover have
    (winding-number p 0 = winding-number q 0) = homotopic-paths (− {0}) p q
    using is-loop-path[OF p(1)] is-loop-path[OF q(1)]
      is-loop-zero-notin-image[OF p(1)] is-loop-zero-notin-image[OF q(1)]
      is-loop-pathstart[OF p(1)] is-loop-pathstart[OF q(1)]
      is-loop-pathfinish[OF p(1)] is-loop-pathfinish[OF q(1)]
    by (intro winding-number-homotopic-paths-eq) simp-all
  ultimately have punct: homotopic-paths (− {0}) p q by simp
  have homotopic-paths circle (retr ∘ p) (retr ∘ q)
   by (rule homotopic-paths-continuous-image[OF punct continuous-on-retr retr-into-circle])
  moreover have homotopic-paths circle p (retr ∘ p)
    using p(1) by (rule homotopic-paths-retr-self)
  moreover have homotopic-paths circle q (retr ∘ q)
    using q(1) by (rule homotopic-paths-retr-self)
  ultimately have homotopic-paths circle p q
    by (meson homotopic-paths-sym homotopic-paths-trans)
  then have loop-class p = loop-class q
    by (rule loop-class-eqI[OF p(1) q(1)])
  then show A = B
    using p q by simp
qed
```

2.8 The degree is surjective

lemma *is-loop-circlepath*: *is-loop* (*circlepath 0 1*)
  by (*simp add*: *is-loop-def circle-def*)

lemma *wnd-circlepath*: *wnd* (*circlepath 0 1*) = *1*
proof −
  have *winding-number* (*circlepath 0 1*) *0* = *1*
    by (*simp add*: *winding-number-circlepath*)
  moreover have *winding-number* (*circlepath 0 1*) *0* = *of-int* (*wnd* (*circlepath 0 1*))
    using *is-loop-circlepath* **by** (*simp add*: *wnd-eq*)
  ultimately show *?thesis* **by** *simp*
qed

The *n*-fold concatenation of the standard circle loop.

fun *cpow* :: *nat* ⇒ (*real* ⇒ *complex*) **where**
  cpow 0 = *linepath 1 1*
| *cpow* (*Suc n*) = *circlepath 0 1* +++ *cpow n*

lemma *is-loop-cpow*: *is-loop* (*cpow n*)
proof (*induction n*)
  case *0*
  show *?case* **unfolding** *cpow.simps*(*1*) **by** (*rule is-loop-linepath-1*)
next
  case (*Suc n*)
  then show *?case* **by** (*simp add*: *is-loop-circlepath*)
qed

lemma *wnd-cpow*: *wnd* (*cpow n*) = *int n*
proof (*induction n*)
  case *0*
  have *wnd* (*cpow 0*) = *0* **unfolding** *cpow.simps*(*1*) **by** (*rule wnd-linepath-1*)
  then show *?case* **by** (*simp del*: *cpow.simps*)
next
  case (*Suc n*)
  have *wnd* (*cpow* (*Suc n*)) = *wnd* (*circlepath 0 1*) + *wnd* (*cpow n*)
    by (*simp add*: *wnd-join is-loop-circlepath is-loop-cpow*)
  also have ... = *int* (*Suc n*)
    by (*simp add*: *wnd-circlepath Suc.IH*)
  finally show *?case* **.**
qed

lemma *deg-surj*: *deg* ‘ *carrier pi1-circle* = *UNIV*
proof −
  have *surj*: ∃ *A* ∈ *carrier pi1-circle*. *deg A* = *n* **for** *n* :: *int*
  proof (*cases n* ≥ *0*)
    case *True*
    have *deg* (*loop-class* (*cpow* (*nat n*))) = *int* (*nat n*)
      by (*simp add*: *deg-class is-loop-cpow wnd-cpow*)

```
    also have ... = n using True by simp
    finally show ?thesis
      using is-loop-cpow by (metis carrier-pi1 loop-class-in-carrier)
  next
    case False
    let ?p = reversepath (cpow (nat (− n)))
    have loop: is-loop ?p
      by (simp add: is-loop-cpow)
    have deg (loop-class ?p) = wnd ?p
      using loop by (simp add: deg-class)
    also have ... = − int (nat (− n))
      by (simp add: wnd-reversepath is-loop-cpow wnd-cpow)
    also have ... = n using False by simp
    finally show ?thesis
      using loop by (metis carrier-pi1 loop-class-in-carrier)
  qed
  show ?thesis
    using surj by (metis UNIV-eq-I image-eqI)
qed
```

2.9 The fundamental theorem

```
theorem pi1-circle-iso-integers: pi1-circle ≅ integer-group
proof −
  have deg ∈ iso pi1-circle integer-group
  proof (rule isoI)
    show deg ∈ hom pi1-circle integer-group
      by (rule deg-hom)
    show bij-betw deg (carrier pi1-circle) (carrier integer-group)
      using deg-inj deg-surj by (simp add: bij-betw-def)
  qed
  then show ?thesis
    by (rule is-isoI)
qed

theorem group-hom-deg: group-hom pi1-circle integer-group deg
  by (simp add: group-hom-def group-hom-axioms-def
      group-pi1-circle group-integer-group deg-hom)

end
```

References

[1] A. Hatcher. *Algebraic Topology*. Cambridge University Press, Cambridge, 2002. Available at https://pi.math.cornell.edu/~hatcher/AT/ATpage.html.

[2] J. R. Munkres. *Topology.* Prentice Hall, Upper Saddle River, NJ, 2nd edition, 2000.