Algebraic Invariants of the Circle, Torus, and Punctured Plane

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

November 15, 2026

Abstract

We formalize real path lifts of based loops in the circle and prove that their integer endpoints classify these loops up to path homotopy. For the torus, the two coordinate lifts give a classification by pairs of integers and an explicit fundamental-group isomorphism with $\mathbb{Z} \times \mathbb{Z}$. Every continuous circle self-map acts on first singular homology by multiplication by its winding number. We construct a radial deformation retraction of the punctured complex plane onto the circle and compute the reduced homology of both spaces: it is isomorphic to the additive group of integers in degree one and trivial in every other degree. Neither space is contractible. The development also provides a reusable interface for path lifting in covering spaces.

Contents

1 Introduction **2**

2 Covering-space lifts for the circle **2**

3 Path lifting **3**

4 Generic pointed path lifts **6**

5 Pointed covering spaces **7**

6 Product paths and torus loops **8**

7 The fundamental group of the torus **10**

8 The punctured plane retracts onto the circle **12**

9 Seifert–van Kampen interface **13**

10 Homology of the circle and the punctured plane **14**

11 Brouwer degree **15**

12 Loops as singular cycles **16**

13 Concatenation and addition of cycles **17**

14 Winding number annihilates singular boundaries **18**

15 Winding number and the induced homology map **19**

16 Circle, torus, and punctured-plane invariants: collected results **20**

1 Introduction

Path lifting assigns an integer to each based loop in the circle. This integer classifies loops up to path homotopy and provides two coordinates for the corresponding classification on the torus. Radial retraction connects the circle with the punctured complex plane and transfers the homology computation between them. These constructions follow standard treatments of algebraic topology, such as Hatcher [1]. We build on the AFP circle entry [3] and use the classical Seifert–van Kampen entry [2].

The principal contributions are the real lift-endpoint classification of circle loops, an explicit torus fundamental-group isomorphism with $\mathbb{Z} \times \mathbb{Z}$, and the transfer of reduced homology to the concrete circle and punctured plane. We also prove that the actual map induced on first singular homology by every continuous circle self-map is multiplication by its winding number, in any integer coordinates. The proof represents loops as singular cycles, relates concatenation to addition using a singular triangle, and proves that winding number annihilates singular boundaries. A homotopy through rotations removes the basepoint restriction on the self-map. The generic lifting interface packages the analysis library's results. The Seifert–van Kampen statements are re-exports of the cited AFP entry; the degree laws specialize the homology library to the abstract one-sphere. The concrete induced-map theorem is separate from the library constant `Brouwer_degree2`; no additional identification with that constant, general Hurewicz theorem, or torus homology computation is asserted here.

The covering interface is independent of the circle development. The product, radial-retract, and homology theories may be imported separately; the final theory combines their interfaces.

theory *Circle-Covering*
  imports
    Fundamental-Group-Circle.Fundamental-Group-Circle
    HOL−Analysis.Further-Topology
begin

2 Covering-space lifts for the circle

This theory complements the winding-number proof in the circle entry with the real-lift description of based circle loops. The covering-space API used below is the one from the standard Isabelle analysis library; in particular, the exponential covering of the punctured plane supplies the analytic lifting argument.

definition *circle-cover* :: *real* ⇒ *complex* **where**
  circle-cover t = *circlepath 0 1 t*

lemma *circle-cover-exp*:
  circle-cover t = *exp* (*2* ∗ *of-real pi* ∗ i ∗ *of-real t*)
  ⟨*proof*⟩

lemma *path-circle-cover* [*simp*]: *path circle-cover*
  ⟨*proof*⟩

lemma *circle-cover-into-circle*: *circle-cover* ‘ *UNIV* ⊆ *circle*
  ⟨*proof*⟩

lemma *circle-cover-periodic-int*:
  circle-cover (*t* + *of-int n*) = *circle-cover t*
  ⟨*proof*⟩

lemma *circle-cover-fiber*:
  circle-cover s = *circle-cover t* ⟷ (∃ *n*::*int*. *s* = *t* + *of-int n*)
  ⟨*proof*⟩

lemma *circle-cover-eq-iff*:
circle-cover s = circle-cover t ⟷ s − t ∈ ℤ
⟨*proof*⟩

lemma *circle-cover-one* [*simp*]: *circle-cover 0 = (1*::*complex)*
⟨*proof*⟩

lemma *circle-cover-Arg*:
*z ∈ circle ⟹ circle-cover (Arg z / (2 * pi)) = z*
⟨*proof*⟩

lemma *circle-cover-mem*: *z ∈ circle ⟹ z ∈ circle-cover ‘ UNIV*
⟨*proof*⟩

lemma *circle-cover-range*: *circle-cover ‘ UNIV = circle*
⟨*proof*⟩

lemma *circle-cover-surjective*: *circle-cover ‘ UNIV = circle*
⟨*proof*⟩

3 Path lifting

definition *circle-path-lift* :: (*real ⇒ complex*) ⇒ (*real ⇒ real*) ⇒ *bool* **where**
circle-path-lift p q ⟷
path q ∧ pathstart q = 0 ∧
(∀ t ∈ {0..1}. circle-cover (q t) = p t)

lemma *circle-path-lift-exists*:
assumes *ppath*: *path p*
and *pimage*: *path-image p ⊆ circle*
and *pstart*: *pathstart p = 1*
shows *∃ q. circle-path-lift p q*
⟨*proof*⟩

definition *circle-lift-exp* :: (*real ⇒ real*) ⇒ (*real ⇒ complex*) **where**
circle-lift-exp q t = ı ∗ of-real (2 ∗ pi ∗ q t)

lemma *path-circle-lift-exp*:
assumes *path q*
shows *path (circle-lift-exp q)*
⟨*proof*⟩

lemma *pathstart-circle-lift-exp*:
assumes *pathstart q = 0*
shows *pathstart (circle-lift-exp q) = 0*
⟨*proof*⟩

lemma *circle-lift-exp-map*:
exp (circle-lift-exp q t) = circle-cover (q t)
⟨*proof*⟩

lemma *circle-path-lift-unique*:
assumes *l1*: *circle-path-lift p q1*
and *l2*: *circle-path-lift p q2*
and *ppath*: *path p*
and *pimage*: *path-image p ⊆ circle*
shows *∀ t ∈ {0..1}. q1 t = q2 t*

⟨*proof*⟩

definition *circle-lift* :: (*real* ⇒ *complex*) ⇒ (*real* ⇒ *real*) **where**
circle-lift p = (*SOME q*. *circle-path-lift p q*)

lemma *circle-lift-spec*:
assumes *path p path-image p* ⊆ *circle pathstart p* = *1*
shows *circle-path-lift p* (*circle-lift p*)
⟨*proof*⟩

lemma *circle-lift-unique*:
assumes *l1*: *circle-path-lift p q1*
and *l2*: *circle-path-lift p q2*
and *ppath*: *path p* **and** *pimage*: *path-image p* ⊆ *circle*
shows ∀ *t* ∈ {*0..1*}. *q1 t* = *q2 t*
⟨*proof*⟩

definition *circle-lift-endpoint* :: (*real* ⇒ *complex*) ⇒ *int* **where**
circle-lift-endpoint p =
(*SOME n*::*int*. *circle-lift p 1* = *of-int n*)

lemma *circle-lift-endpoint-spec*:
assumes *loop*: *is-loop p*
shows *circle-lift p 1* = *of-int* (*circle-lift-endpoint p*)
⟨*proof*⟩

lemma *circle-lift-endpoint-integer*:
assumes *is-loop p*
shows *circle-lift p 1* ∈ ℤ
⟨*proof*⟩

lemma *path-add-const*:
fixes *q* :: *real* ⇒ *real* **and** *a* :: *real*
assumes *path q*
shows *path* (λ*t*. *q t* + *a*)
⟨*proof*⟩

lemma *pathstart-add-const*:
fixes *q* :: *real* ⇒ *real* **and** *a* :: *real*
assumes *pathstart q* = *0*
shows *pathstart* (λ*t*. *q t* + *a*) = *a*
⟨*proof*⟩

lemma *circle-path-lift-join*:
assumes *lp*: *circle-path-lift p P*
and *lq*: *circle-path-lift q Q*
and *endpoint*: *P 1* = *of-int n*
shows *circle-path-lift* (*p* +++ *q*) (*P* +++ (λ*t*. *Q t* + *P 1*))
⟨*proof*⟩

lemma *circle-lift-endpoint-join*:
assumes *p*: *is-loop p* **and** *q*: *is-loop q*
shows *circle-lift-endpoint* (*p* +++ *q*) =
circle-lift-endpoint p + *circle-lift-endpoint q*
⟨*proof*⟩

lemma *circle-path-lift-reverse*:
assumes *lp*: *circle-path-lift p P*

and *endpoint*: $P\ 1 = of\text{-}int\ n$
shows *circle-path-lift* (*reversepath p*) ($\lambda t.\ P\ (1 - t) - P\ 1$)
⟨*proof*⟩

lemma *circle-lift-endpoint-reverse*:
assumes *loop*: *is-loop p*
shows *circle-lift-endpoint* (*reversepath p*) =
$-$ *circle-lift-endpoint p*
⟨*proof*⟩

lemma *continuous-on-circle-cover*: *continuous-on UNIV circle-cover*
⟨*proof*⟩

lemma *circle-lift-endpoint-homotopic*:
assumes *p*: *is-loop p* **and** *q*: *is-loop q*
and *hom*: *homotopic-paths circle p q*
shows *circle-lift-endpoint p* = *circle-lift-endpoint q*
⟨*proof*⟩

lemma *circle-lift-endpoint-eq-homotopic*:
assumes *p*: *is-loop p* **and** *q*: *is-loop q*
and *eq*: *circle-lift-endpoint p* = *circle-lift-endpoint q*
shows *homotopic-paths circle p q*
⟨*proof*⟩

theorem *circle-lift-endpoint-eq-iff*:
assumes *p*: *is-loop p* **and** *q*: *is-loop q*
shows *circle-lift-endpoint p* = *circle-lift-endpoint q* ⟷
homotopic-paths circle p q
⟨*proof*⟩

lemma *circle-path-lift-circlepath*:
circle-path-lift (*circlepath 0 1*) *id*
⟨*proof*⟩

lemma *circle-lift-endpoint-circlepath*:
circle-lift-endpoint (*circlepath 0 1*) = *1*
⟨*proof*⟩

lemma *circle-path-lift-linepath*:
circle-path-lift (*linepath* (*1*::*complex*) *1*) ($\lambda t.\ 0$)
⟨*proof*⟩

lemma *circle-lift-endpoint-linepath*:
circle-lift-endpoint (*linepath* (*1*::*complex*) *1*) = *0*
⟨*proof*⟩

lemma *circle-lift-endpoint-cpow*:
circle-lift-endpoint (*cpow n*) = *int n*
⟨*proof*⟩

lemma *circle-lift-endpoint-eq-wnd*:
assumes *loop*: *is-loop p*
shows *circle-lift-endpoint p* = *wnd p*
⟨*proof*⟩

lemma *circle-lift-endpoint-eq-deg*:
assumes *is-loop p*

shows *circle-lift-endpoint p = deg (loop-class p)*
⟨*proof*⟩

end
theory *Covering-Spaces*
imports *HOL−Analysis.Homeomorphism*
begin

4 Generic pointed path lifts

The standard analysis library contains the point-set definition of a covering space and its lifting theorems. This theory packages those results in a small interface that can be reused without exposing the implementation of a particular covering map. The domain and codomain of the covering are real-normed vector spaces, exactly as required by the analytic covering-space API.

definition *covering-path-lift* ::
'a::*real-normed-vector set* ⇒
(*'a* ⇒ *'b*::*real-normed-vector*) ⇒
(*real* ⇒ *'b*) ⇒ *'a* ⇒ (*real* ⇒ *'a*) ⇒ *bool*
where
covering-path-lift C p g a h ⟷
path h ∧ *path-image h* ⊆ *C* ∧ *pathstart h* = *a* ∧
(∀ *t* ∈ {*0..1*}. *p* (*h t*) = *g t*)

lemma *covering-path-liftI*:
assumes *path h path-image h* ⊆ *C pathstart h* = *a*
and ⋀*t*. *t* ∈ {*0..1*} ⟹ *p* (*h t*) = *g t*
shows *covering-path-lift C p g a h*
⟨*proof*⟩

lemma *covering-path-liftD*:
assumes *covering-path-lift C p g a h*
shows *path h path-image h* ⊆ *C pathstart h* = *a*
and ⋀*t*. *t* ∈ {*0..1*} ⟹ *p* (*h t*) = *g t*
⟨*proof*⟩

lemma *covering-path-lift-exists*:
assumes *cov*: *covering-space C p S* **and** *ac*: *a* ∈ *C*
and *pg*: *path g* **and** *ig*: *path-image g* ⊆ *S*
and *start*: *pathstart g* = *p a*
shows ∃ *h*. *covering-path-lift C p g a h*
⟨*proof*⟩

lemma *covering-path-lift-unique*:
assumes *cov*: *covering-space C p S*
and *pg*: *path g* **and** *ig*: *path-image g* ⊆ *S*
and *h1*: *covering-path-lift C p g a h1*
and *h2*: *covering-path-lift C p g a h2*
shows ⋀*t*. *t* ∈ {*0..1*} ⟹ *h1 t* = *h2 t*
⟨*proof*⟩

definition *covering-path-lift-choice* ::
'a::*real-normed-vector set* ⇒
(*'a* ⇒ *'b*::*real-normed-vector*) ⇒
(*real* ⇒ *'b*) ⇒ *'a* ⇒ (*real* ⇒ *'a*)
where
covering-path-lift-choice C p g a = (*SOME h*. *covering-path-lift C p g a h*)

lemma *covering-path-lift-choice-spec*:
assumes *covering-space C p S a* ∈ *C path g*
path-image g ⊆ *S pathstart g* = *p a*
shows *covering-path-lift C p g a* (*covering-path-lift-choice C p g a*)
⟨*proof*⟩

lemma *covering-path-lift-choice-unique*:
assumes *cov*: *covering-space C p S* **and** *ac*: *a* ∈ *C*
and *pg*: *path g* **and** *ig*: *path-image g* ⊆ *S*
and *start*: *pathstart g* = *p a*
and *h*: *covering-path-lift C p g a h*
shows ⋀*t*. *t* ∈ {*0..1*} ⟹
covering-path-lift-choice C p g a t = *h t*
⟨*proof*⟩

definition *covering-lift-endpoint* ::
'a::*real-normed-vector set* ⇒
(*'a* ⇒ *'b*::*real-normed-vector*) ⇒
(*real* ⇒ *'b*) ⇒ *'a* ⇒ *'a*
where
covering-lift-endpoint C p g a = *covering-path-lift-choice C p g a 1*

lemma *covering-lift-endpoint-eq*:
assumes *cov*: *covering-space C p S* **and** *ac*: *a* ∈ *C*
and *pg*: *path g* **and** *ig*: *path-image g* ⊆ *S*
and *start*: *pathstart g* = *p a*
and *h*: *covering-path-lift C p g a h*
shows *covering-lift-endpoint C p g a* = *h 1*
⟨*proof*⟩

lemma *covering-lift-endpoint-fiber*:
assumes *cov*: *covering-space C p S* **and** *ac*: *a* ∈ *C*
and *pg*: *path g* **and** *ig*: *path-image g* ⊆ *S*
and *start*: *pathstart g* = *p a*
shows *covering-lift-endpoint C p g a* ∈ *C*
and *p* (*covering-lift-endpoint C p g a*) = *pathfinish g*
⟨*proof*⟩

lemma *covering-lift-endpoint-homotopic*:
assumes *cov*: *covering-space C p S*
and *pg1*: *path g1* **and** *ig1*: *path-image g1* ⊆ *S*
and *pg2*: *path g2* **and** *ig2*: *path-image g2* ⊆ *S*
and *hom*: *homotopic-paths S g1 g2*
and *ac*: *a* ∈ *C*
and *start1*: *pathstart g1* = *p a* **and** *start2*: *pathstart g2* = *p a*
shows *covering-lift-endpoint C p g1 a* =
covering-lift-endpoint C p g2 a
⟨*proof*⟩

5 Pointed covering spaces

A pointed covering packages the sheet chosen above the base point. The locale is deliberately lightweight: later developments can instantiate it with the exponential circle cover, universal covers, or covering graphs.

locale *pointed-covering-space* =
fixes *C* :: *'a*::*real-normed-vector set*
and *p* :: *'a* ⇒ *'b*::*real-normed-vector*

```
    and S :: 'b set
    and a0 :: 'a
  assumes covering: covering-space C p S
    and basepoint: a0 ∈ C
begin

lemma continuous:
  continuous-on C p
  ⟨proof⟩

lemma surjective:
  p ‘ C = S
  ⟨proof⟩

abbreviation lift-endpoint where
  lift-endpoint g ≡ covering-lift-endpoint C p g a0

lemma lift-endpoint-homotopic:
  assumes path g1 path-image g1 ⊆ S
    path g2 path-image g2 ⊆ S
    homotopic-paths S g1 g2
    pathstart g1 = p a0 pathstart g2 = p a0
  shows lift-endpoint g1 = lift-endpoint g2
  ⟨proof⟩

end

end
theory Circle-Product
  imports Circle-Covering Covering-Spaces
begin
```

6 Product paths and torus loops

The product construction is the first genuinely functorial extension of the circle calculation. We keep the path-level presentation explicit: this makes the two integer coordinates available independently of the quotient presentation constructed in the torus fundamental-group theory.

```
definition product-path ::
  (real ⇒ 'a::topological-space) ⇒
   (real ⇒ 'b::topological-space) ⇒
   (real ⇒ 'a × 'b)
where
  product-path p q = (λt. (p t, q t))

lemma path-product-path:
  assumes path p path q
  shows path (product-path p q)
  ⟨proof⟩

lemma path-image-product-path:
  path-image (product-path p q) ⊆ path-image p × path-image q
  ⟨proof⟩

lemma pathstart-product-path [simp]:
  pathstart (product-path p q) = (pathstart p, pathstart q)
  ⟨proof⟩
```

lemma *pathfinish-product-path* [*simp*]:
pathfinish (*product-path p q*) = (*pathfinish p*, *pathfinish q*)
⟨*proof*⟩

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

lemma *torus-loopD*:
assumes *torus-loop p*
shows *path p path-image p* ⊆ *circle* × *circle*
pathstart p = (*1*, *1*) *pathfinish p* = (*1*, *1*)
⟨*proof*⟩

lemma *torus-loop-fst*:
assumes *torus-loop p*
shows *is-loop* (*fst* ∘ *p*)
⟨*proof*⟩

lemma *torus-loop-snd*:
assumes *torus-loop p*
shows *is-loop* (*snd* ∘ *p*)
⟨*proof*⟩

lemma *torus-loop-product*:
assumes *is-loop p is-loop q*
shows *torus-loop* (*product-path p q*)
⟨*proof*⟩

definition *torus-lift-endpoint* ::
(*real* ⇒ *complex* × *complex*) ⇒ *int* × *int*
where
torus-lift-endpoint p =
(*circle-lift-endpoint* (*fst* ∘ *p*),
circle-lift-endpoint (*snd* ∘ *p*))

lemma *torus-lift-endpoint-fst* [*simp*]:
fst (*torus-lift-endpoint p*) = *circle-lift-endpoint* (*fst* ∘ *p*)
⟨*proof*⟩

lemma *torus-lift-endpoint-snd* [*simp*]:
snd (*torus-lift-endpoint p*) = *circle-lift-endpoint* (*snd* ∘ *p*)
⟨*proof*⟩

lemma *torus-loop-join*:
assumes *torus-loop p torus-loop q*
shows *torus-loop* (*p* +++ *q*)
⟨*proof*⟩

lemma *torus-loop-reverse*:
assumes *torus-loop p*
shows *torus-loop* (*reversepath p*)
⟨*proof*⟩

lemma *product-path-join*:
product-path (*p* +++ *r*) (*q* +++ *s*) =

product-path p q +++ product-path r s
⟨*proof*⟩

lemma *product-path-reverse*:
product-path (*reversepath p*) (*reversepath q*) =
reversepath (*product-path p q*)
⟨*proof*⟩

lemma *torus-lift-endpoint-join*:
assumes *torus-loop p torus-loop q*
shows *torus-lift-endpoint* (*p +++ q*) =
torus-lift-endpoint p + *torus-lift-endpoint q*
⟨*proof*⟩

lemma *torus-lift-endpoint-reverse*:
assumes *torus-loop p*
shows *torus-lift-endpoint* (*reversepath p*) =
− *torus-lift-endpoint p*
⟨*proof*⟩

lemma *homotopic-paths-product*:
fixes *S* :: *'a*::*topological-space set* **and** *T* :: *'b*::*topological-space set*
and *p q* :: *real* ⇒ *'a* **and** *r s* :: *real* ⇒ *'b*
assumes *h1*: *homotopic-paths S p q*
and *h2*: *homotopic-paths T r s*
shows *homotopic-paths* (*S* × *T*)
(*product-path p r*) (*product-path q s*)
⟨*proof*⟩

lemma *torus-loop-homotopic-iff*:
assumes *p*: *torus-loop p* **and** *q*: *torus-loop q*
shows *homotopic-paths* (*circle* × *circle*) *p q* ⟷
torus-lift-endpoint p = *torus-lift-endpoint q*
⟨*proof*⟩

end
theory *Torus-Fundamental-Group*
imports *Circle-Torus-Algebraic-Topology.Circle-Product*
begin

7 The fundamental group of the torus

Elements are homotopy classes of loops based at (*1*,*1*). Multiplication is induced by path concatenation. The two lift endpoints give an explicit isomorphism to the product of the additive integer groups.

definition *torus-class* **where**
torus-class p = {*q*. *torus-loop q* ∧ *homotopic-paths* (*circle* × *circle*) *p q*}

definition *torus-classes* **where**
torus-classes = {*torus-class p* |*p*. *torus-loop p*}

definition *torus-rep* **where**
torus-rep A = (*SOME p*. *torus-loop p* ∧ *torus-class p* = *A*)

lemma *torus-class-mem*:
torus-loop p ⟹ *torus-class p* ∈ *torus-classes*
⟨*proof*⟩

lemma *torus-class-self*:
torus-loop p $\Longrightarrow$ *p* $\in$ *torus-class p*
⟨*proof*⟩

lemma *torus-class-eq-iff*:
assumes *torus-loop p torus-loop q*
shows *torus-class p* = *torus-class q* $\longleftrightarrow$
torus-lift-endpoint p = *torus-lift-endpoint q*
⟨*proof*⟩

lemma *torus-rep-props*:
assumes *A* $\in$ *torus-classes*
shows *torus-loop* (*torus-rep A*) $\wedge$ *torus-class* (*torus-rep A*) = *A*
⟨*proof*⟩

definition *torus-degree* **where**
torus-degree A = *torus-lift-endpoint* (*torus-rep A*)

lemma *torus-degree-class*:
assumes *torus-loop p*
shows *torus-degree* (*torus-class p*) = *torus-lift-endpoint p*
⟨*proof*⟩

lemma *torus-degree-inj*: *inj-on torus-degree torus-classes*
⟨*proof*⟩

lemma *torus-endpoint-product*:
torus-lift-endpoint (*product-path p q*) =
(*circle-lift-endpoint p*, *circle-lift-endpoint q*)
⟨*proof*⟩

lemma *circle-endpoint-surj*:
obtains *p* **where** *is-loop p circle-lift-endpoint p* = *n*
⟨*proof*⟩

lemma *torus-degree-surj*: *torus-degree* ‘ *torus-classes* = *UNIV*
⟨*proof*⟩

definition *pi1-torus* :: (*real* $\Rightarrow$ *complex* $\times$ *complex*) *set monoid* **where**
pi1-torus = (|*carrier* = *torus-classes*,
mult = (λ*A B*. *torus-class* (*torus-rep A* +++ *torus-rep B*)),
one = *torus-class* (*product-path* (*linepath 1 1*) (*linepath 1 1*))|)

lemma *torus-mult-class*:
assumes *torus-loop p torus-loop q*
shows *torus-class p* $\otimes_{pi1\text{-}torus}$ *torus-class q* = *torus-class* (*p* +++ *q*)
⟨*proof*⟩

lemma *torus-mult-closed*:
assumes *A* $\in$ *torus-classes B* $\in$ *torus-classes*
shows *A* $\otimes_{pi1\text{-}torus}$ *B* $\in$ *torus-classes*
⟨*proof*⟩

lemma *torus-degree-mult*:
assumes *A* $\in$ *torus-classes B* $\in$ *torus-classes*
shows *torus-degree* (*A* $\otimes_{pi1\text{-}torus}$ *B*) = *torus-degree A* + *torus-degree B*
⟨*proof*⟩

lemma *torus-one-closed*: $\mathbf{1}_{pi1\text{-}torus} \in torus\text{-}classes$
⟨*proof*⟩

lemma *torus-degree-one*: $torus\text{-}degree\ \mathbf{1}_{pi1\text{-}torus} = 0$
⟨*proof*⟩

theorem *group-pi1-torus*: *group pi1-torus*
⟨*proof*⟩

lemma *torus-inverse-class*:
assumes *torus-loop p*
shows $inv_{pi1\text{-}torus}\ (torus\text{-}class\ p) = torus\text{-}class\ (reversepath\ p)$
⟨*proof*⟩

theorem *torus-degree-iso*:
torus-degree ∈ *iso pi1-torus*
(*Fundamental-Group-Circle.integer-group* ×× *Fundamental-Group-Circle.integer-group*)
⟨*proof*⟩

theorem *pi1-torus-iso-integer-product*:
pi1-torus ≅
(*Fundamental-Group-Circle.integer-group* ×× *Fundamental-Group-Circle.integer-group*)
⟨*proof*⟩

end
theory *Circle-Radial-Retract*
imports *Circle-Covering*
begin

8 The punctured plane retracts onto the circle

The preceding circle entry already contains the analytic radial map *retr*. We expose it here with the geometric name used by the later homotopy and homology developments, together with its subspace-topology interface.

definition *punctured-plane* :: *complex set* **where**
punctured-plane = − {0}

definition *radial-retraction* :: *complex* ⇒ *complex* **where**
radial-retraction z = *retr z*

lemma *punctured-plane-eq*:
punctured-plane = − {0}
⟨*proof*⟩

lemma *radial-retraction-continuous*:
continuous-on punctured-plane radial-retraction
⟨*proof*⟩

lemma *radial-retraction-into-circle*:
radial-retraction ∈ *punctured-plane* → *circle*
⟨*proof*⟩

lemma *radial-retraction-fixes*:
assumes *z* ∈ *circle*
shows *radial-retraction z* = *z*
⟨*proof*⟩

lemma *radial-retraction-maps*:
retraction-maps (*top-of-set punctured-plane*) (*top-of-set circle*)
radial-retraction id
⟨*proof*⟩

definition *radial-homotopy* :: *real* × *complex* ⇒ *complex* **where**
radial-homotopy z =
((1 − *fst z*) ∗ (1 / *norm* (*snd z*)) + *fst z*) ∗$_R$ *snd z*

lemma *radial-homotopy-coeff-pos*:
assumes *t* ∈ {0..1} *z* ∈ *punctured-plane*
shows 0 < (1 − *t*) ∗ (1 / *norm z*) + *t*
⟨*proof*⟩

lemma *radial-homotopy-into-punctured-plane*:
radial-homotopy ∈ ({0..1} × *punctured-plane*) → *punctured-plane*
⟨*proof*⟩

lemma *radial-homotopy-continuous*:
continuous-on ({0..1} × *punctured-plane*) *radial-homotopy*
⟨*proof*⟩

lemma *radial-homotopy-continuous-map*:
continuous-map (*prod-topology* (*top-of-set* {0..1}) (*top-of-set punctured-plane*))
(*top-of-set punctured-plane*) *radial-homotopy*
⟨*proof*⟩

lemma *radial-homotopy-zero*:
radial-homotopy (0,*z*) = *radial-retraction z*
⟨*proof*⟩

lemma *radial-homotopy-one*:
radial-homotopy (1,*z*) = *id z*
⟨*proof*⟩

lemma *radial-deformation*:
homotopic-with (λ*f*. *True*) (*top-of-set punctured-plane*) (*top-of-set punctured-plane*)
radial-retraction id
⟨*proof*⟩

lemma *punctured-plane-homotopy-equivalent-circle*:
top-of-set punctured-plane homotopy-equivalent-space top-of-set circle
⟨*proof*⟩

end
theory *Circle-SVK*
imports *Circle-Radial-Retract Seifert−Van−Kampen.Seifert-Van-Kampen*
begin

9 Seifert–van Kampen interface

The substantial open-cover construction lives in the AFP *Seifert−Van−Kampen* entry. This theory makes that result a named layer of this development, so later circle and punctured-plane applications can depend on the established carrier-level fundamental-group interface.

context *classical-svk-setup*
begin

theorem *svk-classical-interface*:
bij-betw classical-svk-quotient-map (*fundamental-group-space W x0*)
(*carrier-full-amalgamated-free-product-space G1 G2 H i1 i2 mult1 one1 mult2 one2*)
⟨*proof*⟩

end

context *topological-svk*
begin

interpretation *svk-carrier-interface*:
carrier-svk-encode-decode-interface
G1 G2 H i1 i2 mult1 one1 mult2 one2 encode svk-decode
⟨*proof*⟩

theorem *svk-topological-interface*:
bij-betw svk-carrier-interface.carrier-svk-quotient-map UNIV
(*carrier-full-amalgamated-free-product-space G1 G2 H i1 i2 mult1 one1 mult2 one2*)
⟨*proof*⟩

end

end
theory *Circle-Homology-Degree*
imports *Circle-Radial-Retract HOL−Homology.Homology*
begin

10 Homology of the circle and the punctured plane

The standard Isabelle homology library computes the reduced homology of an abstract sphere. The geometric circle used by the preceding theories is identified with the one-sphere using the library's Euclidean-coordinate homeomorphism, and the radial deformation retract then transfers the result to the punctured plane.

lemma *reduced-homology-group-nsphere-one*:
reduced-homology-group (*int* (*1*::*nat*)) (*nsphere* (*1*::*nat*)) ≅ *integer-group*
⟨*proof*⟩

lemma *reduced-homology-group-nsphere-one-other*:
p ≠ *int* (*1*::*nat*) ⟹
trivial-group (*reduced-homology-group p* (*nsphere* (*1*::*nat*)))
⟨*proof*⟩

lemma *circle-homeomorphic-nsphere-one*:
top-of-set circle homeomorphic-space nsphere 1
⟨*proof*⟩

lemma *reduced-homology-group-circle-one*:
reduced-homology-group (*int* (*1*::*nat*)) (*top-of-set circle*) ≅ *integer-group*
⟨*proof*⟩

lemma *reduced-homology-group-circle-transfer*:
reduced-homology-group p (*top-of-set circle*) ≅
reduced-homology-group p (*nsphere* (*1*::*nat*))
⟨*proof*⟩

lemma *reduced-homology-group-circle-other*:

$p \neq int\ (1{::}nat) \Longrightarrow$
$trivial\text{-}group\ (reduced\text{-}homology\text{-}group\ p\ (top\text{-}of\text{-}set\ circle))$
⟨*proof*⟩

lemma *reduced-homology-group-punctured-plane-transfer*:
$reduced\text{-}homology\text{-}group\ p\ (top\text{-}of\text{-}set\ punctured\text{-}plane) \cong$
$reduced\text{-}homology\text{-}group\ p\ (top\text{-}of\text{-}set\ circle)$
⟨*proof*⟩

lemma *reduced-homology-group-punctured-plane-one*:
$reduced\text{-}homology\text{-}group\ (int\ (1{::}nat))\ (top\text{-}of\text{-}set\ punctured\text{-}plane) \cong integer\text{-}group$
⟨*proof*⟩

lemma *reduced-homology-group-punctured-plane-other*:
$p \neq int\ (1{::}nat) \Longrightarrow$
$trivial\text{-}group\ (reduced\text{-}homology\text{-}group\ p\ (top\text{-}of\text{-}set\ punctured\text{-}plane))$
⟨*proof*⟩

lemma *circle-not-contractible*:
$\neg\ contractible\text{-}space\ (top\text{-}of\text{-}set\ circle)$
⟨*proof*⟩

lemma *punctured-plane-not-contractible*:
$\neg\ contractible\text{-}space\ (top\text{-}of\text{-}set\ punctured\text{-}plane)$
⟨*proof*⟩

11 Brouwer degree

The homology library defines the degree of a self-map of an abstract sphere through its action on the top reduced homology group. We expose the basic calculational laws needed by later degree arguments.

lemma *Brouwer-degree-circle-identity*:
$Brouwer\text{-}degree2\ 1\ id = 1$
⟨*proof*⟩

lemma *Brouwer-degree-circle-reflection*:
$Brouwer\text{-}degree2\ 1\ (\lambda x\ i.\ if\ i = 0\ then\ -x\ i\ else\ x\ i) = -1$
⟨*proof*⟩

lemma *Brouwer-degree-circle-compose*:
assumes $continuous\text{-}map\ (nsphere\ 1)\ (nsphere\ 1)\ f$
and $continuous\text{-}map\ (nsphere\ 1)\ (nsphere\ 1)\ g$
shows $Brouwer\text{-}degree2\ 1\ (g \circ f) =$
$Brouwer\text{-}degree2\ 1\ g * Brouwer\text{-}degree2\ 1\ f$
⟨*proof*⟩

lemma *Brouwer-degree-circle-homotopy-invariant*:
assumes $homotopic\text{-}with\ (\lambda x.\ True)\ (nsphere\ 1)\ (nsphere\ 1)\ f\ g$
shows $Brouwer\text{-}degree2\ 1\ f = Brouwer\text{-}degree2\ 1\ g$
⟨*proof*⟩

end
theory *Circle-Loop-Homology*
imports *Circle-Torus-Algebraic-Topology.Circle-Covering*
Circle-Torus-Algebraic-Topology.Circle-Homology-Degree
begin

12 Loops as singular cycles

definition *edge-coordinates* :: *real* ⇒ *nat* ⇒ *real* **where**
edge-coordinates t i = (*if i* = *0 then* 1−*t else if i* = *1 then t else* 0)

lemma *edge-coordinates-mem*:
t ∈ {*0..1*} ⟹ *edge-coordinates t* ∈ *standard-simplex 1*
⟨*proof*⟩

lemma *edge-coordinates-continuous*:
continuous-map euclideanreal (*powertop-real UNIV*) *edge-coordinates*
⟨*proof*⟩

definition *loop-simplex* :: (*real* ⇒ *complex*) ⇒ (*nat* ⇒ *real*) ⇒ *complex* **where**
loop-simplex p = *restrict* (λ*x*. *p* (*x 1*)) (*standard-simplex 1*)

lemma *loop-simplex-singular*:
assumes *path p path-image p* ⊆ *S*
shows *singular-simplex 1* (*top-of-set S*) (*loop-simplex p*)
⟨*proof*⟩

lemma *loop-simplex-edge*:
t ∈ {*0..1*} ⟹ *loop-simplex p* (*edge-coordinates t*) = *p t*
⟨*proof*⟩

lemma *loop-simplex-boundary*:
chain-boundary 1 (*frag-of* (*loop-simplex p*)) =
frag-of (*restrict* (λ*x*. *p 1*) (*standard-simplex 0*)) −
frag-of (*restrict* (λ*x*. *p 0*) (*standard-simplex 0*))
⟨*proof*⟩

lemma *loop-simplex-cycle*:
assumes *is-loop p*
shows *singular-relcycle 1* (*top-of-set circle*) {} (*frag-of* (*loop-simplex p*))
⟨*proof*⟩

definition *circle-loop-homology* **where**
circle-loop-homology p = *homologous-rel-set 1* (*top-of-set circle*) {} (*frag-of* (*loop-simplex p*))

lemma *circle-loop-homology-carrier*:
is-loop p ⟹ *circle-loop-homology p* ∈ *carrier* (*homology-group 1* (*top-of-set circle*))
⟨*proof*⟩

lemma *loop-simplex-compose*:
simplex-map 1 f (*loop-simplex p*) = *loop-simplex* (*f* ∘ *p*)
⟨*proof*⟩

lemma *circle-loop-homology-natural*:
assumes *continuous-map* (*top-of-set circle*) (*top-of-set circle*) *f is-loop p*
shows *hom-induced 1* (*top-of-set circle*) {} (*top-of-set circle*) {} *f*
(*circle-loop-homology p*) = *circle-loop-homology* (*f* ∘ *p*)
⟨*proof*⟩

definition *circle-loop-map* **where**
circle-loop-map p = *p* ∘ (λ*z*. *Arg2pi z* / (*2*∗*pi*))

lemma *circle-loop-map-homotopic*:
assumes *homotopic-loops circle p q*

shows *homotopic-with* (λ*h*. *True*) (*top-of-set circle*) (*top-of-set circle*)
(*circle-loop-map p*) (*circle-loop-map q*)
⟨*proof*⟩

lemma *circle-loop-map-continuous*:
assumes *is-loop p*
shows *continuous-map* (*top-of-set circle*) (*top-of-set circle*) (*circle-loop-map p*)
⟨*proof*⟩

lemma *circle-loop-map-parametrize*:
assumes *is-loop p t* ∈ {*0..1*}
shows *circle-loop-map p* (*circlepath 0 1 t*) = *p t*
⟨*proof*⟩

lemma *circle-loop-homology-map*:
assumes *is-loop p*
shows *hom-induced 1* (*top-of-set circle*) {} (*top-of-set circle*) {} (*circle-loop-map p*)
(*circle-loop-homology* (*circlepath 0 1*)) = *circle-loop-homology p*
⟨*proof*⟩

lemma *circle-loop-homology-homotopic*:
assumes *is-loop p is-loop q homotopic-paths circle p q*
shows *circle-loop-homology p* = *circle-loop-homology q*
⟨*proof*⟩

end
theory *Circle-Loop-Concatenation*
imports *Circle-Loop-Homology*
begin

13 Concatenation and addition of cycles

lemma *homology-class-add*:
homologous-rel-set 1 X {} (*a*+*b*) =
homologous-rel-set 1 X {} *a* ⊗$_{homology\text{-}group\ 1\ X}$ *homologous-rel-set 1 X* {} *b*
⟨*proof*⟩

definition *join-simplex* **where**
join-simplex p q = *restrict* (λ*x*. (*p* +++ *q*) (*x 1* / *2* + *x 2*)) (*standard-simplex 2*)

lemma *join-simplex-singular*:
assumes *is-loop p is-loop q*
shows *singular-simplex 2* (*top-of-set circle*) (*join-simplex p q*)
⟨*proof*⟩

lemma *join-simplex-faces*:
assumes *is-loop p is-loop q*
shows *singular-face 2 0* (*join-simplex p q*) = *loop-simplex q*
singular-face 2 1 (*join-simplex p q*) = *loop-simplex* (*p* +++ *q*)
singular-face 2 2 (*join-simplex p q*) = *loop-simplex p*
⟨*proof*⟩

lemma *circle-loop-homology-join*:
assumes *is-loop p is-loop q*
shows *circle-loop-homology* (*p* +++ *q*) =
circle-loop-homology p ⊗$_{homology\text{-}group\ 1\ (top\text{-}of\text{-}set\ circle)}$ *circle-loop-homology q*
⟨*proof*⟩

lemma *circle-loop-homology-constant*:
circle-loop-homology (*linepath 1 1*) = $\mathbf{1}_{\textit{homology-group 1 (top-of-set circle)}}$
⟨*proof*⟩

lemma *circle-loop-homology-reverse*:
assumes *is-loop p*
shows *circle-loop-homology* (*reversepath p*) =
$\textit{inv}_{\textit{homology-group 1 (top-of-set circle)}}$ (*circle-loop-homology p*)
⟨*proof*⟩

lemma *circle-loop-homology-cpow*:
circle-loop-homology (*Fundamental-Group-Circle.cpow n*) =
circle-loop-homology (*circlepath 0 1*) $[\widehat{\ }]_{\textit{homology-group 1 (top-of-set circle)}}$ *n*
⟨*proof*⟩

theorem *circle-loop-homology-winding*:
assumes *is-loop p*
shows *circle-loop-homology p* =
circle-loop-homology (*circlepath 0 1*) $[\widehat{\ }]_{\textit{homology-group 1 (top-of-set circle)}}$ *wnd p*
⟨*proof*⟩

end
theory *Circle-Winding-Cocycle*
imports *Circle-Loop-Homology*
begin

14 Winding number annihilates singular boundaries

The winding functional is extended linearly to singular chains. The boundary of each singular triangle has winding zero, because its boundary loop contracts in the triangle. This detects a nonzero first homology class without assuming a Hurewicz theorem.

definition *winding-triangle* :: *complex set* **where**
winding-triangle = {*z*. $0 \le Re\ z \wedge 0 \le Im\ z \wedge Re\ z + Im\ z \le 1$}

lemma *convex-winding-triangle*: *convex winding-triangle*
⟨*proof*⟩

definition *triangle-coordinates* :: *complex* ⇒ *nat* ⇒ *real* **where**
triangle-coordinates z i =
(*if i=0 then 1 − Re z − Im z else if i=1 then Re z else if i=2 then Im z else 0*)

lemma *triangle-coordinates-mem*:
assumes *z* ∈ *winding-triangle*
shows *triangle-coordinates z* ∈ *standard-simplex 2*
⟨*proof*⟩

lemma *triangle-coordinates-continuous*:
continuous-map euclidean (*powertop-real UNIV*) *triangle-coordinates*
⟨*proof*⟩

lemma *singular-triangle-continuous*:
assumes *singular-simplex 2* (*top-of-set circle*) *f*
shows *continuous-on winding-triangle* (*f* ∘ *triangle-coordinates*)
(*f* ∘ *triangle-coordinates*) ‘ *winding-triangle* ⊆ *circle*
⟨*proof*⟩

lemma *triangle-winding-add*:
assumes *cont*: *continuous-on winding-triangle f* **and** *im*: *f ‘ winding-triangle* $\subseteq$ *circle*
shows *winding-number* (*f* $\circ$ *linepath 1* i) *0* $-$
winding-number (*f* $\circ$ *linepath 0* i) *0* $+$
winding-number (*f* $\circ$ *linepath 0 1*) *0* $=$ *0*
⟨*proof*⟩

lemma *singular-triangle-winding*:
assumes *singular-simplex 2* (*top-of-set circle*) *f*
shows *winding-number* (*singular-face 2 0 f* $\circ$ *edge-coordinates*) *0* $-$
winding-number (*singular-face 2 1 f* $\circ$ *edge-coordinates*) *0* $+$
winding-number (*singular-face 2 2 f* $\circ$ *edge-coordinates*) *0* $=$ *0*
⟨*proof*⟩

definition *complex-add-group* :: *complex monoid* **where**
complex-add-group $=$ ⦇*carrier* $=$ *UNIV*, *mult* $=$ (+), *one* $=$ *0*⦈

lemma *comm-group-complex-add*: *comm-group complex-add-group*
⟨*proof*⟩

lemma *winding-chain-functional*:
obtains *w* :: *complex chain* $\Rightarrow$ *complex* **where**
w 0 $=$ *0* $\bigwedge a\ b.\ w\ (a+b) = w\ a + w\ b$ $\bigwedge a\ b.\ w\ (a-b) = w\ a - w\ b$
$\bigwedge f.\ w\ (\mathit{frag\text{-}of}\ f) =$ *winding-number* (*f* $\circ$ *edge-coordinates*) *0*
$\bigwedge c.$ *singular-chain 2* (*top-of-set circle*) *c* $\Longrightarrow$ *w* (*chain-boundary 2 c*) $=$ *0*
⟨*proof*⟩

theorem *circle-standard-loop-homology-nontrivial*:
circle-loop-homology (*circlepath 0 1*) $\neq \mathbf{1}_{\textit{homology-group 1 (top-of-set circle)}}$
⟨*proof*⟩

end
theory *Circle-Homology-Action*
imports *Circle-Winding-Cocycle Circle-Loop-Concatenation*
begin

15 Winding number and the induced homology map

An endomorphism of an infinite cyclic group is determined by its action on any nonzero element. Thus the nonzero standard-loop class and the loop winding formula suffice; we need not first prove that this class is a primitive generator. The final theorem applies to arbitrary continuous circle maps, using a rotation homotopy to normalize the basepoint.

lemma *infinite-cyclic-hom-determined*:
assumes *grp*: *group G* **and** *iso*: *h* $\in$ *iso G Elementary-Groups.integer-group*
and *hom*: *F* $\in$ *hom G G*
and *a*: *a* $\in$ *carrier G a* $\neq \mathbf{1}_G$
and *image*: *F a* $=$ *a* $[\widehat{}]_G$ (*n*::*int*)
and *x*: *x* $\in$ *carrier G*
shows *h* (*F x*) $=$ *n* $*$ *h x*
⟨*proof*⟩

lemma *circle-first-homology-iso*:
homology-group 1 (*top-of-set circle*) $\cong$ *Elementary-Groups.integer-group*
⟨*proof*⟩

lemma *circle-based-map-loop*:

assumes *continuous-map* (*top-of-set circle*) (*top-of-set circle*) *f f 1 = 1 is-loop p*
shows *is-loop* (*f ∘ p*)
⟨*proof*⟩

theorem *circle-homology-action-winding-based*:
assumes *coordinates*: *h ∈ iso* (*homology-group 1* (*top-of-set circle*)) *Elementary-Groups.integer-group*
and *f*: *continuous-map* (*top-of-set circle*) (*top-of-set circle*) *f f 1 = 1*
and *x*: *x ∈ carrier* (*homology-group 1* (*top-of-set circle*))
shows *h* (*hom-induced 1* (*top-of-set circle*) {} (*top-of-set circle*) {} *f x*) =
wnd (*f ∘ circlepath 0 1*) ∗ *h x*
⟨*proof*⟩

lemma *circle-map-winding-homotopic*:
assumes *homotopic-with-canon* (*λh. True*) *circle circle f g*
shows *wnd* (*f ∘ circlepath 0 1*) = *wnd* (*g ∘ circlepath 0 1*)
⟨*proof*⟩

lemma *circle-map-based-normalization*:
assumes *f*: *continuous-map* (*top-of-set circle*) (*top-of-set circle*) *f*
obtains *g* **where** *continuous-map* (*top-of-set circle*) (*top-of-set circle*) *g g 1 = 1*
homotopic-with-canon (*λh. True*) *circle circle f g*
⟨*proof*⟩

theorem *circle-homology-action-winding*:
assumes *coordinates*: *h ∈ iso* (*homology-group 1* (*top-of-set circle*)) *Elementary-Groups.integer-group*
and *f*: *continuous-map* (*top-of-set circle*) (*top-of-set circle*) *f*
and *x*: *x ∈ carrier* (*homology-group 1* (*top-of-set circle*))
shows *h* (*hom-induced 1* (*top-of-set circle*) {} (*top-of-set circle*) {} *f x*) =
wnd (*f ∘ circlepath 0 1*) ∗ *h x*
⟨*proof*⟩

end
theory *Circle-Torus-Algebraic-Topology*
imports *Torus-Fundamental-Group Circle-SVK Circle-Homology-Action*
begin

16 Circle, torus, and punctured-plane invariants: collected results

This theory collects the circle, torus, and punctured-plane results. Its component theories separate the reusable covering-space API, product and retract constructions, the Seifert–van Kampen interface, and the homology/degree consequences.

theorem *circle-fundamental-group-summary*:
pi1-circle ≅ *Fundamental-Group-Circle.integer-group*
⟨*proof*⟩

theorem *circle-reduced-homology-summary*:
reduced-homology-group 1 (*top-of-set circle*) ≅
Elementary-Groups.integer-group
⟨*proof*⟩

theorem *punctured-plane-homology-and-noncontractibility*:
reduced-homology-group 1 (*top-of-set punctured-plane*) ≅ *integer-group* ∧
¬ *contractible-space* (*top-of-set punctured-plane*)
⟨*proof*⟩

end

References

[1] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

[2] A. F. Ramos, D. B. Hulak, and R. J. G. B. de Queiroz. The classical Seifert–van Kampen theorem. *Archive of Formal Proofs*, May 2026. `https://isa-afp.org/entries/Seifert-Van-Kampen.html`, Formal proof development.

[3] A. F. Ramos, D. B. Hulak, and R. J. G. B. de Queiroz. The fundamental group of the circle. *Archive of Formal Proofs*, July 2026. `https://isa-afp.org/entries/Fundamental_Group_Circle.html`, Formal proof development.