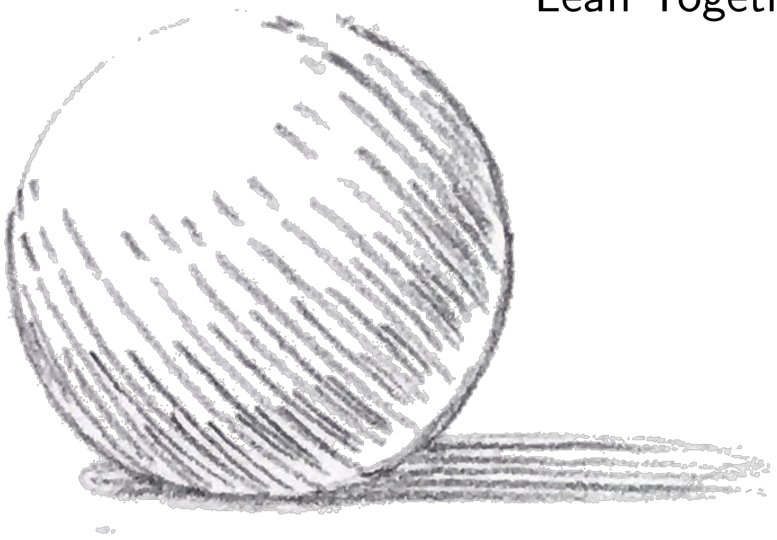
An example of a manifold

Heather Macbeth

Fordham University

Lean Together 2021



Summary

Project: The sphere is a smooth manifold.

Scale:

- $\sim$ 400 lines of code refactor, $\sim$ 500 new lines of code in preliminaries
- $\sim$ 300 lines of code for the result

Status: 8 merged PRs to `mathlib`, 6 open PRs

Foundations: `mathlib` libraries for smooth functions (`times_cont_diff`) and manifolds, the work of Sébastien Gouëzel

This has been formalized before . . .

Smooth Manifolds and Types to Sets for Linear Algebra in Isabelle/HOL

Fabian Immler
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, USA
fimmler@cs.cmu.edu

Bohua Zhan
State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences
Beijing, China
bzhan@ios.ac.cn

Abstract

We formalize the definition and basic properties of smooth manifolds in Isabelle/HOL. Concepts covered include partition of unity, tangent and cotangent spaces, and the fundamental theorem for line integrals. We also construct some concrete manifolds such as spheres and projective spaces. The formalization makes extensive use of the existing libraries for topology and analysis. The existing library for linear algebra is not flexible enough for our needs. We therefore set up the first systematic and large scale application of “types to sets”. It allows us to automatically transform the existing (type based) library of linear algebra to one with explicit carrier sets.

CCS Concepts • **Theory of computation → Higher order logic**; *Logic and verification*; • **Mathematics of computing** → *Geometric topology*.

Keywords Isabelle, Higher Order Logic, Manifolds, Formalization of Mathematics

ACM Reference Format:
Fabian Immler and Bohua Zhan. 2019. Smooth Manifolds and Types to Sets for Linear Algebra in Isabelle/HOL. In *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP ’19), January 14–15, 2019, Cascais, Portugal.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3293880.3294093

1 Introduction

geometry (leading to the modern theory of classical mechanics). It also plays important roles in the theory of dynamical systems and partial differential equations. Formalization of the theory of smooth manifolds in a proof assistant, therefore, is an important step towards making interactive theorem proving applicable to many areas of study.

In addition to its importance in mathematics, formalizing smooth manifolds is also interesting as a difficult test case for proof assistants. Reasoning about smooth manifolds requires large libraries in both mathematical analysis and linear algebra. Moreover, the prevalent use of subsets and partial functions, as well as constructions depending on dimension or points in the manifold, offer a rigorous test of the proof assistant’s type system.

In this paper, we describe how to formalize the basic concepts of smooth manifolds in Isabelle/HOL. We largely follow chapters 1, 2, 3, and 11 of the textbook *Introduction to Smooth Manifolds* by Lee [12], formalizing about one half of the material in these chapters. Occasionally, we also refer to other textbooks such as [5] and [19].

Our developments are available in the Archive of Formal Proof [9] and consist of about 11k lines of code.

We emphasize that we are formalizing in this paper manifolds with a smooth structure, not just topological manifolds, a simpler concept that has already been formalized in systems such as Mizar [16]. Moreover, we treat manifolds as abstract topological spaces endowed with compatible charts,

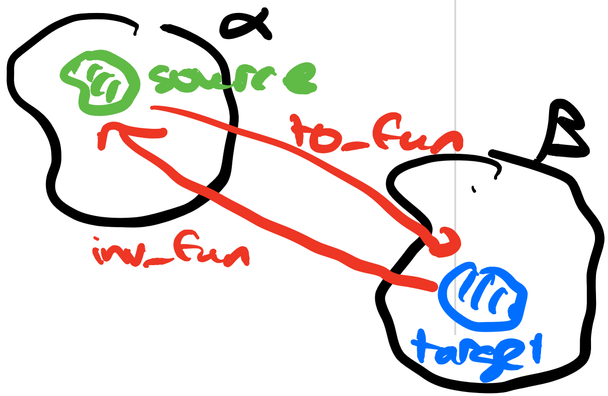
This has been formalized before . . .

```
proof goal_cases
  case 1
  have *: "smooth_on ((\<lambda>(x::'a, z). x /\<^sub>R (1 - z)) ` (({a. norm a = 1} - {(0, 1)}) \<inter> ({a. norm a = 1} - {(0, - 1)})))
    ((\<lambda>(x, z). x /\<^sub>R (1 + z)) \<circ> (\<lambda>x. ((2 / ((norm x)\<^sup>2 + 1)) *\<^sub>R x, ((norm x)\<^sup>2 - 1) / ((norm x)\<^sup>2 + 1))
    apply (rule smooth_on_subset[where T="UNIV - {0}"])
    subgoal
      by (auto intro!: smooth_on_divide smooth_on_inverse smooth_on_scaleR smooth_on_mult smooth_on_add
          smooth_on_minus smooth_on_norm simp: o_def power2_eq_square add_nonneg_eq_0_iff divide_simps)
    apply (auto simp: norm_prod_def power2_eq_square) apply sos
    done
  show ?case
    by transfer (rule *)
next
  case 2
  have *: "smooth_on ((\<lambda>(x::'a, z). x /\<^sub>R (1 + z)) ` (({a. norm a = 1} - {(0, 1)}) \<inter> ({a. norm a = 1} - {(0, - 1)})))
    ((\<lambda>(x, z). x /\<^sub>R (1 - z)) \<circ> (\<lambda>x. ((2 / ((norm x)\<^sup>2 + 1)) *\<^sub>R x, (1 - (norm x)\<^sup>2) / ((norm x)\<^sup>2 + 1))
    apply (rule smooth_on_subset[where T="UNIV - {0}"])
    subgoal
      by (auto intro!: smooth_on_divide smooth_on_inverse smooth_on_scaleR smooth_on_mult smooth_on_add
          smooth_on_minus smooth_on_norm simp: o_def power2_eq_square add_nonneg_eq_0_iff divide_simps)
    apply (auto simp: norm_prod_def add_eq_0_iff) apply sos
    done
  show ?case
    by transfer (rule *)
qed

definition charts_sphere :: "('a::euclidean_space sphere, 'a) chart set" where
  "charts_sphere \<equiv> {st_proj1_chart, st_proj2_chart}"

lemma c_manifold_atlas_sphere: "c_manifold charts_sphere \<infinity>"
  apply (unfold_locales)
  unfolding charts_sphere_def
  using smooth_compat_commute smooth_compat_refl st_projs_compat by fastforce

end
```

Let's take a tour of the `mathlib` manifolds library, using the sphere as our example.

Step 1: Local homeomorphisms

```
/-- Local equivalence between subsets `source` and `target` of α and β respectively. The (global)
maps `to_fun : α → β` and `inv_fun : β → α` map `source` to `target` and conversely, and are inverse
to each other there. The values of `to_fun` outside of `source` and of `inv_fun` outside of `target`
are irrelevant. -/
@[nolint has_inhabited_instance]
structure local_equiv (α : Type*) (β : Type*) :=
(to_fun      : α → β)
(inv_fun     : β → α)
(source      : set α)
(target      : set β)
(map_source' : ∀{x}, x ∈ source → to_fun x ∈ target)
(map_target' : ∀{x}, x ∈ target → inv_fun x ∈ source)
(left_inv'   : ∀{x}, x ∈ source → inv_fun (to_fun x) = x)
(right_inv'  : ∀{x}, x ∈ target → to_fun (inv_fun x) = x)
```




```
/-- local homeomorphisms, defined on open subsets of the space -/
@[nolint has_inhabited_instance]
structure local_homeomorph (α : Type*) (β : Type*) [topological_space α] [topological_space β]
  extends local_equiv α β :=
(open_source        : is_open source)
(open_target        : is_open target)
(continuous_to_fun  : continuous_on to_fun source)
(continuous_inv_fun : continuous_on inv_fun target)
```

Step 1: Local homeomorphisms

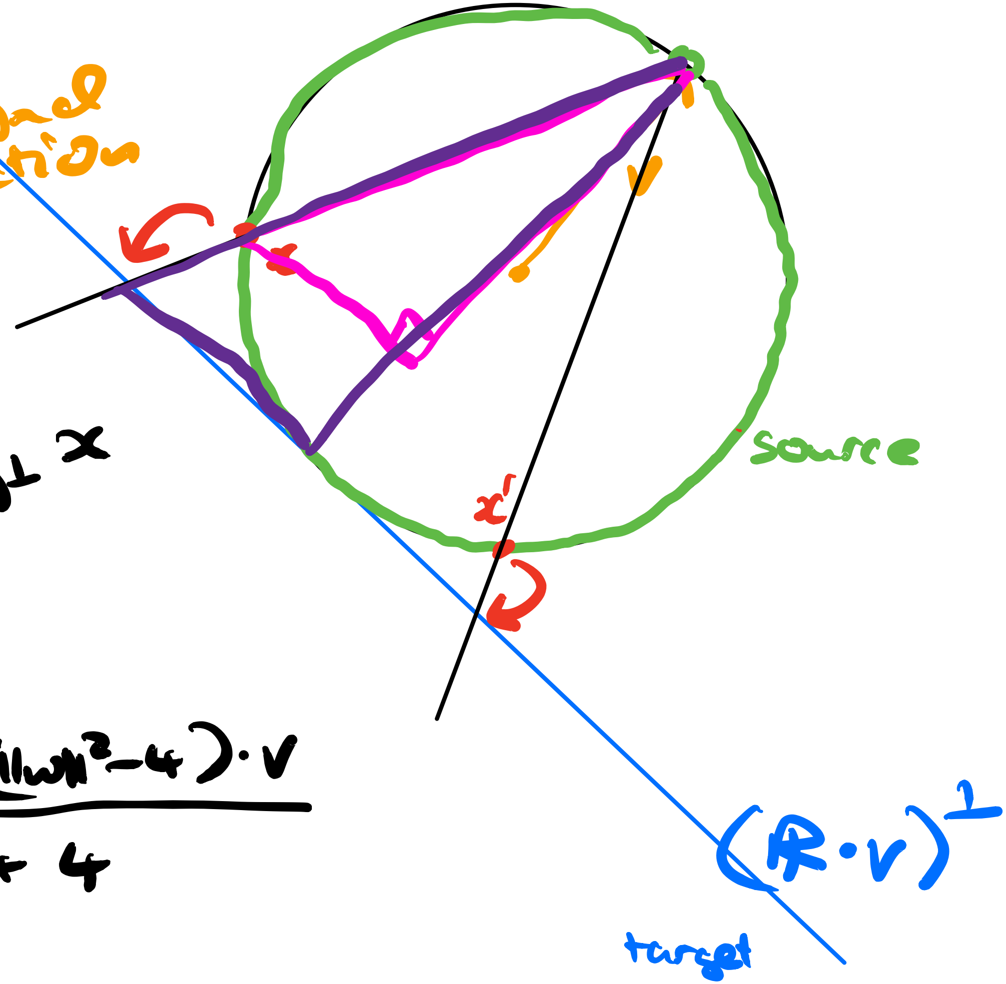
to-fun:

orthogonal projection

$$x \longmapsto \frac{2}{1-\langle v, x\rangle} \cdot P_{(\mathbb{R}\cdot v)^{\perp}}\, x$$

inv-fun:

$$w \longmapsto \frac{4w + (\|w\|^2-4)\cdot v}{\|w\|^2+4}$$



Step 1: Local homeomorphisms

Check are left & right inverse to each other.

For $x \in$ sphere, write as $a \cdot v + w$, $w \perp v$.

$$\frac{4\left(\frac{2w}{1-a}\right) + \left[\left\|\frac{2w}{1-a}\right\|^2 - 4\right] \cdot v}{\left\|\frac{2w}{1-a}\right\|^2 + 4}$$

$$= \frac{4(1-a) \cdot 2w + \left[4\|w\|^2 - 4(1-a)^2\right] \cdot v}{4\|w\|^2 + 4(1-a)^2}$$

$= \cdots = a \cdot v + w$, given $a^2 + \|w\|^2 = 1$.

Step 1: Local homeomorphisms

```
      orthogonal_projection_mem_subspace_eq_self w,
    have h₃ : inner_right v w = (0:ℝ) := inner_right_of_mem_orthogonal_singleton v w.2,
    have h₄ : inner_right v v = (1:ℝ) := by simp [real_inner_self_eq_norm_square, hv],
    simp [h₁, h₂, h₃, h₄, continuous_linear_map.map_add, continuous_linear_map.map_smul,
      mul_smul] },
  { simp }
end

/-- Stereographic projection from the unit sphere in `E`, centred at a unit vector `v` in `E`; this
is the version as a local homeomorphism. -/
def stereographic (hv : ‖v‖ = 1) : local_homeomorph (sphere (0:E) 1) (ℝ ∙ v)ᗮ :=
{ to_fun := (stereo_to_fun v) ∘ coe,
  inv_fun := stereo_inv_fun hv,
  source := {⟨v, by simp [hv]⟩}ᶜ,
  target := set.univ,
  map_source' := by simp,
  map_target' := λ w _, stereo_inv_fun_ne_north_pole hv w,
  left_inv' := λ _ hx, stereo_left_inv hv (λ h, hx (subtype.ext h)),
  right_inv' := λ w _, stereo_right_inv hv w,
  open_source := is_open_compl_singleton,
  open_target := is_open_univ,
  continuous_to_fun := continuous_on_stereo_to_fun.comp continuous_subtype_coe.continuous_on
    (λ w h, h ∘ subtype.ext ∘ eq.symm ∘ (inner_eq_norm_mul_iff_of_norm_one hv (by simp)).mp),
  continuous_inv_fun := (continuous_stereo_inv_fun hv).continuous_on }

@[simp] lemma stereographic_source (hv : ‖v‖ = 1) :
  (stereographic hv).source = {⟨v, by simp [hv]⟩}ᶜ :=
rfl

@[simp] lemma stereographic_target (hv : ‖v‖ = 1) : (stereographic hv).target = set.univ := rfl
```

Step 1: Local homeomorphisms

```
lemma stereo_left_inv (hv : ‖v‖ = 1) {x : sphere (0:E) 1} (hx : (x:E) ≠ v) :
  stereo_inv_fun hv (stereo_to_fun v x) = x :=
begin
  ext,
  simp only [stereo_to_fun_apply, stereo_inv_fun_apply, smul_add],
  -- name two frequently-occuring quantities and write down their basic properties
  set a : ℝ := inner_right v x,
  set y := orthogonal_projection (ℝ ∙ v)ᗮ x,
  have split : ↑x = a • v + ↑y,
  { convert eq_sum_orthogonal_projection_self_orthogonal_complement (ℝ ∙ v) x,
    exact (orthogonal_projection_unit_singleton ℝ hv x).symm },
  have hvy : ⟪v, y⟫_ℝ = 0 := inner_right_of_mem_orthogonal_singleton v y.2,
  have pythag : 1 = a ^ 2 + ‖(y:E)‖ ^ 2,
  { have hvy' : ⟪a • v, y⟫_ℝ = 0 := by simp [inner_smul_left, hvy],
    convert norm_add_square_eq_norm_square_add_norm_square_of_inner_eq_zero _ _ hvy' using 2,
    { simp [← split] },
    { simp [norm_smul, hv, real.norm_eq_abs, ← pow_two, abs_sq_eq] },
    { exact pow_two _ } },
  -- two facts which will be helpful for clearing denominators in the main calculation
  have ha : 1 - a ≠ 0,
  { have : a < 1 := (inner_lt_one_iff_real_of_norm_one hv (by simp)).mpr hx.symm,
    linarith },
  have : 2 ^ 2 * ‖(y:E)‖ ^ 2 + 4 * (1 - a) ^ 2 ≠ 0,
  { refine ne_of_gt _,
    have := norm_nonneg (y:E),
    have : 0 < (1 - a) ^ 2 := pow_two_pos_of_ne_zero (1 - a) ha,
    nlinarith },
  -- the core of the problem is these two algebraic identities:
  have h₁ : (2 ^ 2 / (1 - a) ^ 2 * ‖y‖ ^ 2 + 4)⁻¹ * 4 * (2 / (1 - a)) = 1,
  { field_simp,
```

Step 2: Charted space ($\approx$ topological manifold)

```
/-! ### Charted spaces -/
/-- A charted space is a topological space endowed with an atlas, i.e., a set of local
homeomorphisms taking value in a model space `H`, called charts, such that the domains of the charts
cover the whole space. We express the covering property by chosing for each `x` a member
`chart_at H x` of the atlas containing `x` in its source: in the smooth case, this is convenient to
construct the tangent bundle in an efficient way.
The model space is written as an explicit parameter as there can be several model spaces for a
given topological space. For instance, a complex manifold (modelled over `ℂ^n`) will also be seen
sometimes as a real manifold over `ℝ^(2n)`.
-/
class charted_space (H : Type*) [topological_space H] (M : Type*) [topological_space M] :=
(atlas []            : set (local_homeomorph M H))
(chart_at []         : M → local_homeomorph M H)
(mem_chart_source [] : ∀x, x ∈ (chart_at x).source)
(chart_mem_atlas []  : ∀x, chart_at x ∈ atlas)

export charted_space
attribute [simp, mfld_simps] mem_chart_source chart_mem_atlas

section charted_space

/-- Any space is a charted_space modelled over itself, by just using the identity chart -/
instance charted_space_self (H : Type*) [topological_space H] : charted_space H H :=
{ atlas            := {local_homeomorph.refl H},
  chart_at         := λx, local_homeomorph.refl H,
  mem_chart_source := λx, mem_univ x,
  chart_mem_atlas  := λx, mem_singleton _ }

/-- In the trivial charted_space structure of a space modelled over itself through the identity, the
```

Step 2: Charted space ($\approx$ topological manifold)

We have, for each $v \in$ sphere,
a local homeomorphism
sphere $\to (\mathbb{R} \cdot v)^\perp$
(source $= \{v\}^c \subseteq$ sphere, target = univ)

Need to identify the $(\mathbb{R} \cdot v)^\perp$

Method 1 Fix $v \in$ sphere. Use only charts for v and $-v$.

Method 2 Fix arbitrary isomorphisms $(\mathbb{R} \cdot v)^\perp \xrightarrow{\sim} \mathbb{R}^n$.

Step 2: Charted space (≈ topological manifold)

```
orthogonalization, but in the finite-dimensional case it follows more easily by dimension-counting.
-/

/-- Variant of the stereographic projection (see `stereographic`), for the sphere in a finite-
dimensional inner product space `E`.  This version has codomain the Euclidean space of dimension
`findim ℝ E - 1`. -/
def stereographic' (v : sphere (0:E) 1) :
  local_homeomorph (sphere (0:E) 1) (euclidean_space ℝ (fin (findim ℝ E - 1))) :=
(stereographic (norm_eq_of_mem_sphere v)).trans
(continuous_linear_equiv.of_findim_eq
( begin
    rw findim_orthogonal_span_singleton (nonzero_of_mem_unit_sphere v),
    simp
  end )).to_homeomorph.to_local_homeomorph

@[simp] lemma stereographic'_source (v : sphere (0:E) 1) :
  (stereographic' v).source = {v}ᶜ :=
by simp [stereographic']

@[simp] lemma stereographic'_target (v : sphere (0:E) 1) :
  (stereographic' v).target = set.univ :=
by simp [stereographic']

/-- The unit sphere in a finite-dimensional inner product space `E` is a charted space modelled on
the Euclidean space of dimension `findim ℝ E - 1`. -/
instance : charted_space (euclidean_space ℝ (fin (findim ℝ E - 1))) (sphere (0:E) 1) :=
{ atlas            := { f | ∃ v : (sphere (0:E) 1), f = stereographic' v},
  chart_at         := λ v, stereographic' (-v),
  mem_chart_source := λ v, by simpa using ne_neg_of_mem_unit_sphere ℝ v,
  chart_mem_atlas  := λ v, ⟨-v, rfl⟩ }
```

Step 3: Smooth manifold

```
/-! ### Smooth manifolds with corners -/

set_option old_structure_cmd true

/-- Typeclass defining smooth manifolds with corners with respect to a model with corners, over a
field `𝕜` and with infinite smoothness to simplify typeclass search and statements later on. -/
@[ancestor has_groupoid]
class smooth_manifold_with_corners {𝕜 : Type*} [nondiscrete_normed_field 𝕜]
  {E : Type*} [normed_group E] [normed_space 𝕜 E]
  {H : Type*} [topological_space H] (I : model_with_corners 𝕜 E H)
  (M : Type*) [topological_space M] [charted_space H M] extends
  has_groupoid M (times_cont_diff_groupoid ∞ I) : Prop

lemma smooth_manifold_with_corners_of_times_cont_diff_on
  {𝕜 : Type*} [nondiscrete_normed_field 𝕜]
  {E : Type*} [normed_group E] [normed_space 𝕜 E]
  {H : Type*} [topological_space H] (I : model_with_corners 𝕜 E H)
  (M : Type*) [topological_space M] [charted_space H M]
  (h : ∀ (e e' : local_homeomorph M H), e ∈ atlas H M → e' ∈ atlas H M →
    times_cont_diff_on □ □ (I ∘ (e.symm □□ e') ∘ I.symm)
      (I.symm ⁻¹' (e.symm □□ e').source ∩ range I)) :
  smooth_manifold_with_corners I M :=
{ compatible :=
  begin
    haveI : has_groupoid M (times_cont_diff_groupoid ∞ I) := has_groupoid_of_pregroupoid _ h,
    apply structure_groupoid.compatible,
  end }

/-- For any model with corners, the model space is a smooth manifold -/
instance model_space_smooth {𝕜 : Type*} [nondiscrete_normed_field 𝕜]
```

Step 3: Smooth manifold

```
lemma times_cont_diff_on_stereo_to_fun [complete_space E] :
  times_cont_diff_on ℝ ⊤ (stereo_to_fun v) {x : E | inner_right v x ≠ (1:ℝ)} :=
begin
  refine times_cont_diff_on.smul _
    (orthogonal_projection ((ℝ ∙ v)ᗮ)).times_cont_diff.times_cont_diff_on,
  refine times_cont_diff_const.times_cont_diff_on.div _ _,
  { exact (times_cont_diff_const.sub (inner_right v).times_cont_diff).times_cont_diff_on },
  { intros x h h',
    exact h (sub_eq_zero.mp h').symm }
end
```

```
lemma times_cont_diff_stereo_inv_fun_aux : times_cont_diff ℝ ⊤ (stereo_inv_fun_aux v) :=
begin
  have h₀ : times_cont_diff ℝ ⊤ (λ w : E, ‖w‖ ^ 2) := times_cont_diff_norm_square,
  have h₁ : times_cont_diff ℝ ⊤ (λ w : E, (‖w‖ ^ 2 + 4)⁻¹),
  { refine (h₀.add times_cont_diff_const).inv _,
    intros x,
    nlinarith },
  have h₂ : times_cont_diff ℝ ⊤ (λ w, (4:ℝ) • w + (‖w‖ ^ 2 - 4) • v),
  { refine (times_cont_diff_const.smul times_cont_diff_id).add _,
    refine (h₀.sub times_cont_diff_const).smul times_cont_diff_const },
  convert h₁.smul h₂
end
```

Preliminaries

- Refactor orthogonal projection library:
  - upgrade from `linear_map` to `continuous_linear_map`
  - change standard hypothesis from `(h : is_complete (K : set E))` to `[complete_space K]`
  - change codomain from `E` (whole space) to `K` (subspace)
- Notation for orthogonal complement $K^\perp$ and span of a single vector $\mathbb{R} \cdot v$
- Fill gaps in the library:
  - Orthogonal complement of the orthogonal complement is itself
  - Span of a single nonzero vector has dimension one
  - Two unit vectors have inner product < 1, if and only if they are different
  - etc etc
- Practically no new lemmas needed about continuity or smoothness. Maybe these libraries are complete?

Further exercises

- Make some smooth maps:
  - the covering map : $\mathbb{R} \to S^1$
  - the Hopf fibration : $S^3 \to S^2$
- Put a conformally flat structure on the sphere (i.e., define the conformally flat groupoid, show that the transition functions belong to it)
- Make other manifolds:
  - projective space
  - $SO(n)$
  - level sets of submersions
  - orbits of free proper Lie group actions
- Define submanifolds and quotient manifolds. Show that the sphere is a submanifold of $\mathbb{R}^n$.
- Sphere eversion