Topology Filters - Notes

Release 0.1

Carlos Caralps

Sep 22, 2021

CONTENTS

CHAPTER

ONE

FILTER DEFINITION AND ALGEBRAIC STRUCTURE

We will start defining filters and, then the elementary filter propositions will be proved by the usual way and by Lean. This chapter aims to define an algebraic structure with filters using two operations.

1.1 Filter definition

Firstly, we will introduce the filter definition of a giving set.

Definition 1.1.1 (Filter). *Let* `X` *be a set, a filter is a family of subsets of the power ser* `F ⊆ 𝒫(X)` *satisfying the next properties*

(i) *The universal set is in the filter* `X ∈ F`.

(ii) *If* `E ∈ F`, *then* `∀A ∈ 𝒫(X)` *such that* `E ⊆ A`, *we have* `A ∈ F`.

(iii) *If* `E, A ∈ F`, *then* `E ∩ A ∈ F`.

The reader might have noticed we have not included the empty axiom (states that the empty set cannot be in any filter) commonly used in filter definitions and required for topology filter convergence. Assuming it, would make it impossible to define the neutral element in one of the operations we will use later.

Having the conceptual definition of filters, we can define this structure in Lean. The following code lines were published in the mathlib repository, being the current definition of filters on that repository.

```
structure filter (X : Type) :=
(sets                   : set (set X))
(univ_sets              : set.univ ∈ sets)
(sets_of_superset {x y} : x ∈ sets → x ⊆ y → y ∈ sets)
(inter_sets {x y}       : x ∈ sets → y ∈ sets → x ∩ y ∈ sets)
```

Having introduced the definition of filters, we will proceed with defining the principal filters. Those are essential to lots of topological structures as the open neighbourhood of a point.

Definition 1.1.2 (Principal Filter). *Let* `X` *a set and* `A ⊆ X` *a subset. We define the principal filter as the subset* `{t ∈ 𝒫(X) | s ⊆ t}`, *and from now onwards, it will be denoted as* `P(A)`.

We have introduced a definition of what we have supposed to be a particular type of filter. Now, we should prove that it fulfils the conditions for being a filter.

Proposition 1.1.3 *Let* `X` *a set. For all* `A ⊆ X` *subsets, the principal filter of* `A` *is a filter.*

Proof. We will prove that a principal filter is a filter by proving the three properties of filters.

(i) It is clear that `A ⊆ X`. Then, by definition, we have `X ∈ P(A)`.

(ii) If we have `E ∈ P(A)`, by definition, we also have `A ⊆ E`. For all `B ∈ 𝒫(X)` such that `E ⊆ B`, we will have `A ⊆ B` because of fundamental set propositions. Then we can conclude that `B ∈ P(A)`.

(iii) If we have `E, B ∈ P(A)`, by definition, we will have `A ⊆ E` and `A ⊆ B`. Because `A` is contained in both subsets, we also have `A ⊆ E ∩ B`, which led us to `E ∩ B ∈ P(A)`. ∎

When we attend to define a principal filter in Lean, we will be required to prove that this object is a filter. The following lines are from mathlib repository, being the definition for principal filters that Lean community uses.

```
import data.set.basic
open set

structure filter (X : Type) :=
(sets                    : set (set X))
(univ_sets               : set.univ ∈ sets)
(sets_of_superset {x y} : x ∈ sets → x ⊆ y → y ∈ sets)
(inter_sets {x y}        : x ∈ sets → y ∈ sets → x ∩ y ∈ sets)

def principal {X : Type} (s : set X) : filter X :=
{ sets               := {t | s ⊆ t},
  univ_sets          := subset_univ s,
  sets_of_superset   := assume x y hx hy, subset.trans hx hy,
  inter_sets         := assume x y, subset_inter }

localized "notation `P` := filter.principal" in filter
```

1.2 Filter Order

1.3 Exercices

This subsection aims to propose some exercises that will help the reader to test the knowledge presented above. All are written in Lean and the usual way and separated into the sections we have followed.

1.3.1 Filter definition

(i) **Exercise 1.** Let `X` be a set, a filter `F` of `X` and two subsets `V, U ⊆ X`. The intersection of the subsets is on the filter if only if both are in the filter.

(ii) **Exercise 2.** Let `X` be a set, a filter `F` of `X` and two subsets `V, U ⊆ X`. If the subset `{x | if x ∈ V then x ∈ U }` is in the filter, then `U` is in the filter if `V` is in the filter.

```
import data.set.basic
open set

structure filter (X : Type) :=
(sets                    : set (set X))
(univ_sets               : set.univ ∈ sets)
(sets_of_superset {x y} : x ∈ sets → x ⊆ y → y ∈ sets)
(inter_sets {x y}        : x ∈ sets → y ∈ sets → x ∩ y ∈ sets)

def principal {X : Type} (s : set X) : filter X :=
{ sets               := {t | s ⊆ t},
  univ_sets          := subset_univ s,
  sets_of_superset   := assume x y hx hy, subset.trans hx hy,
  inter_sets         := assume x y, subset_inter }
```

(continues on next page)

(continued from previous page)

```
localized "notation `P` := filter.principal" in filter
variables {X : Type} {f : filter X}

lemma exercise1 {s t} : s ∩ t ∈ f.sets ↔ s ∈ f.sets ∧ t ∈ f.sets :=
begin
  sorry
end

lemma exercise2 {s t} (h : {x | x ∈ s → x ∈ t} ∈ f.sets) :
  s ∈ f.sets → t ∈ f.sets :=
begin
  sorry
end
```