Filters

Definition

A filter `F` on a type `α` is set in `Set α` (*i. e.* a collection of sets in `α`) such that:

1. The largest set `⊤ = Set.univ` is in `F`;
2. If `s,t : Set α` and `s ⊆ t`, then `s ∈ F` implies that `t ∈ F` (they are "upwards closed")
3. `F` is stable by finite intersections.

More precisely, `Filter` is a structure:

```
structure Filter (α : Type*) : Type*
  | sets : Set (Set α)
  | univ_sets : univ ∈ self.sets
  | sets_of_superset : ∀ {x y : Set α}, x ∈ sets → x ⊆ y → y ∈ sets
  | inter_sets : ∀ {x y : Set α}, x ∈ sets → y ∈ sets → x ∩ y ∈ sets
```

- If `F` is a filter on `α`, and `U` is a subset of `α` then we can write `U ∈ F` as on paper, instead of the pedantic `U ∈ F.sets`.

+++ Some examples of filters

- Given a term `a : α`, the collection of all sets containing `a` is the **principal** filter (at `a`): this generalises to any set `S ⊆ α`, being the case `S = {a}`. It is denoted `𝓟 S`, typed `\MCP S`.

⌘

- The collection of all sets of natural integers (or real numbers, or rational numbers...) that are "large enough" or "small enough" are filters. They are called `atTop` and `atBot`, respectively.

- More generally—and this is really the key case connecting these notions with some topology—in a topological space `X`, the collection of all neighbourhoods (or of all open neighbourhoods) of a subspace `S` is a filter, denoted `𝓝 S`. We content ourselves with the case of metric spaces (and of `ℝ`, quite often).

⌘
+++

Why filters

Filters are (among other things) a very convenient way to talk about **convergence**.

Consider a function $f : \mathbb{R} \to \mathbb{R}$ and $a,b : \mathbb{R}$: then
$$
\lim_{x \to a} f (x) = b
$$
means

$$
∀; ε > 0, ∃; δ > 0 ;\text{ such that }; ||x - a|| < δ ⇒ ||f(x) - b|| < ε
$$
or, equivalently,
$$
∀; ε > 0, ∃; δ > 0 ;\text{ such that }; f (a - δ, a + δ) ⊆ (b - ε, b + ε).
$$
This means that for every neighbourhood U_b of b, there exists a neighbourhood V_a of a such that $V_a ⊆ f^{-1}U_b$: since $f^{-1}U_b ∈ 𝓝 b$, upwards-closeness of filters transforms this into

```
∀ U : Set ℝ, U ∈  𝓝 b → f⁻¹' U ∈ 𝓝 a.
```

What about the statement
$$\lim_{x → +∞} f(x)=b\quad ?$$
It simply becomes

```
∀ U : Set ℝ, U ∈  𝓝 b → f⁻¹' U ∈ (atTop : Filter ℝ) .
```

Similarly, if $(a_n)*{n ∈ ℕ}$ is a sequence (here with real values, but it could have values "everywhere"), the statement $$
lim{n → +∞} a_n=b
$$
means that $a : ℕ → ℝ$ converges to $b : ℝ$, thus is equivalent to

```
∀ U : Set ℝ, U ∈  𝓝 b → a⁻¹' U ∈ (atTop : Filter ℕ)
```

meaning that $a^{-1} U$ contains an interval $[n, +∞)$, which is exactly the fact that "for arbitrarily large n, the value a_n is arbitrarily close to b".

- All these definitions of convergence can be written in the same way by using filters. Already *nice*, but really **powerful** when we prove theorems.

For example, let $f,g : ℝ → ℝ$ and $a,b,c ∈ ℝ$. One theorem is that
$$
\lim_{x → a}f (x)=a ⇒ \lim_{y → c}g(y)= c ⇒ \lim_{x → a}(g∘ f)(x)=c
$$
and
$$
\lim_{x → +∞}f (x)=a ⇒ \lim_{y → c}g(y)= c ⇒ \lim_{x → +∞}(g∘ f)(x)=c
$$
is *another* theorem, because $+∞ ∉ ℝ$.

- **On paper**: "the proof is similar".

- **With Lean**: need to rewrite the proof. Consider all possibilities (including limits at infinity, limits as `x` is only in a certain subset etc), and ask yourselves if you really want to write the resulting 3487 lemmas (conservative estimation).

- If instead we **express everything with filters**, then we only need to prove each statement *once*.

+++ *Convergence*, **Take 1**
First attempt to define convergence: `f : α → β` is a function, we have a filter `F` on `α`, a filter `G` on `β`, and we want to say `f` tends to `β` along `α`.
We generalise the definition that appeared before:

```
def Tendsto_preimage (f : α → β) (F : Filter α) (G : Filter β) : Prop :=
  ∀ V ∈ G, f ⁻¹' V ∈ F
```

⌘

A small drawback of the definition `Tendsto_preimage` is that it exposes a quantifier `∀`, and this is

1. Aesthetically unpleasant
2. Slightly cumbersome from the formalisation point of view, since it forces us to constantly use `intro` and to reason "with terms" rather than trying to have a more abstract approach directly working with their types.
   +++

+++ *Convergence*, **Take 2** or: *An intuitive way to think about filters, and a reformulation of convergence*

(*Recall*: For every `s : Set α`, the principal filter `𝓟 s` was the filter whose elements are the sets containing `s`.)

- Think of `𝓟 s` as replacing `s`, and of more general filters as "generalised sets" of `α`. So, for `F : Filter α`, saying `s ∈ F` means that `s` "contains" the corresponding "generalised set".

- Indeed, as we saw when `α = ℝ`, we have `s ∈ 𝓝 a ↔ ∃ ε > 0, ball a ε ⊆ s`. Here, the "generalised set" is an infinitesimal thickening of `{a}` representing arbitrarily small open balls centred at `a`.

- If `α = ℕ`, then `Filter.atTop` is "the set of elements that are large enough".

+++ Filters as generalised sets : NON HO SISTEMATO IL CODICE CHE VA AVEC
Since we're looking at filters as generalised sets, let's extend some set-theoretical notions to them.

1. The **order** relation: sets on `α` are ordered by inclusion, so `T₁ ≤ T₂ ↔ T₁ ⊆ T₂ ↔ ∀ s, s ⊇ T₂ → s ⊇ T₁`. Hence:

```
theorem le_def (F G : Filter α) : F ≤ G ↔ ∀ s ∈ G, s ∈ F := Iff.rfl
```

2. Image of a filter through a function `f : α → β`. This operation is called `Filter.map`, and `Filter.map F f = F.map f` by "dot-notation". We want

   ```
    theorem mem_map (t : Set β) (F : Filter α) : t ∈ Filter.map f F ↔ f ⁻¹' t ∈
   F := Iff.rfl

    theorem mem_map (t : Set β) (F : Filter α) : t ∈ F.map f ↔ f ⁻¹' t ∈ F :=
   Iff.rfl
   ```

Convergence

Given $f : \mathbb{R} \to \mathbb{R}$, we have $\lim_{x \to a}f(x) = b$ if, for every $x \in \mathbb{R}$ close to a, its image $f(x)$ is close to b: in other words f sends the "set of elements close to a" to a "generalised subset" of "the generalised set of elements that are sufficiently close to b": in formulæ,

$$
\lim_{x \to a}f(x) = b \Leftrightarrow (\mathcal{N} a).\mathtt{map}\; f \leq \mathcal{N} b.
$$

All this becomes

```
def Tendsto (f : α → β) (F : Filter α) (G : Filter β) := F.map f ≤ G
```

⌘

+++

Eventually true properties

Filters also allow to talk about properties that are "eventually true": true for large enough `x`, true if `x` is sufficiently close to a fixed point `a`, true for almost all `x` etc.

Given `p: α → Prop` and `F : Filter α`, we have the

```
def F.Eventually p : Prop := {x | p x} ∈ F
```

The notation for this is: `∀ᶠ x in F, p x`: type `∀ᶠ` as `\forall + \^f`.

Intuitively, this means that `p` is true on the "generalised set" corresponding to `F`:

- If `F = atTop`, the statement `{x | p x} ∈ F` means that `p` is true for large enough `x`: and if `F = 𝓝 a`, then `p` is true for all `x` in a neighbourhood of `a`.

- The notation `[=]ᶠ` (**no space** between `=`, `ᶠ` and the limit) is the special case when `p` is an equality: given a filter `F : Filter α`, and two functions `f g : α → β`,

  ```
  f =ᶠ[F] g ↔ ∀ᶠ x in l, f x = g x
  ```

  so `f g` are "eventually equal".

- How to express that a claim is true "for almost all `x`"?

⌘

+++ Axiomatic of filters and `∀ᶠ`

1. `⊤ ∈ F` means that: `∀ x, p x → ∀ᶠ x in F, p x`.

2. The stability of `F` by taking a superset means that, if `q : α → Prop` is another function, and if `∀ᶠ x, p x` and `∀ x, p x → q x`, then `∀ᶠ x, q x`.

3. The stability of `F` by intersections means that, if `∀ᶠ x in F, p x` and `∀ᶠ x in F, q x`, then `∀ᶠ x in F, p x ∧ q x`.

⌘ → Some exercises for you
+++

Frequently true properties

Another filter notion is `Filter.Frequently`. You would use it for example to express something like "there exist arbitrarily large `n` in `ℕ` such that *so-and-so*".

By definition,

```
(∃ᶠ x in F, p x) ↔ (¬ ∀ᶠ x in F, ¬ p x)
```