Workshop on Lean

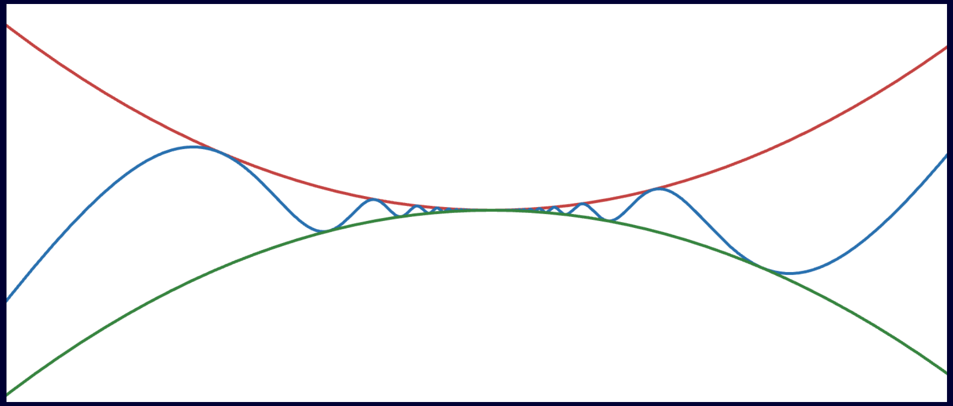
Analysis Lecture 2: "Filters"

Alex Kontorovich

RUTGERS UNIVERSITY / IAS

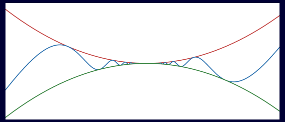
Analysis Lecture 2: “Filters”

- Recall: Squeeze Theorem: Let $a, b, c : \mathbb{N} \to \mathbb{R}$ be sequences with $a \leq b \leq c$ (for all *n*) and $a \to L, c \to L$, then $b \to L$
- Another Squeeze Theorem: Let $a, b, c : \mathbb{R} \to \mathbb{R}$ be functions with $a \leq b \leq c$ (for all x) and $a(x) \to L$, $c(x) \to L$ as $x \to x_0$, then $b(x) \to L$

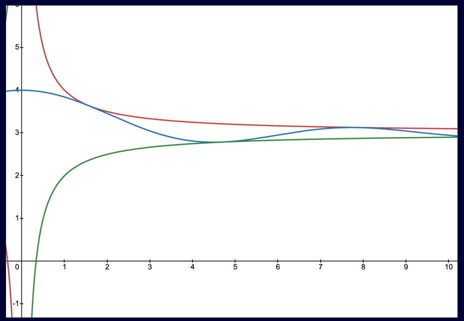


Analysis Lecture 2: “Filters”

- Recall: Squeeze Theorem: Let $a, b, c : \mathbb{N} \to \mathbb{R}$ be sequences with $a \leq b \leq c$ (for all *n*) and $a \to L, c \to L$, then $b \to L$
- Another Squeeze Theorem: Let $a, b, c : \mathbb{R} \to \mathbb{R}$ be functions with $a \leq b \leq c$ (for all x) and $a(x) \to L$, $c(x) \to L$ as $x \to x_0$, then $b(x) \to L$

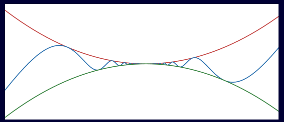


- Another Squeeze Theorem: Let $a, b, c : \mathbb{R} \to \mathbb{R}$ be functions with $a \leq b \leq c$ (for all x) and $a(x) \to L$, $c(x) \to L$ as $x \to \infty$, then $b(x) \to L$

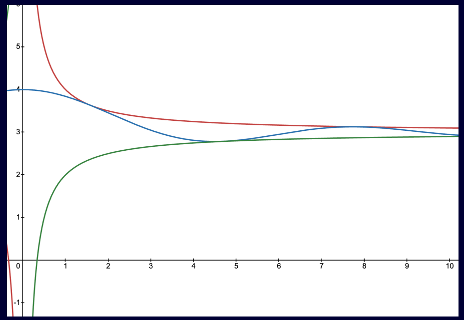


Analysis Lecture 2: “Filters”

- Another Squeeze Theorem: Let $a, b, c : \mathbb{R} \to \mathbb{R}$ be functions with $a \le b \le c$ (for all x) and $a(x) \to L$, $c(x) \to L$ as $x \to x_0$, then $b(x) \to L$



- Another Squeeze Theorem: Let $a, b, c : \mathbb{R} \to \mathbb{R}$ be functions with $a \le b \le c$ (for all x) and $a(x) \to L$, $c(x) \to L$ as $x \to \infty$, then $b(x) \to L$



- Another Squeeze Theorem: Let $a, b, c : \mathbb{R} \to \mathbb{R}$ be functions with $a \le b \le c$ (for all x) and $a(x) \to L$, $c(x) \to L$ as $x \to x_0^+$, then $b(x) \to L$

…

Analysis Lecture 2: “Filters”

- Another Squeeze Theorem: Let $a, b, c : \mathbb{R} \to \mathbb{R}$ be functions with $a \leq b \leq c$ (for all x) and $a(x) \to L$, $c(x) \to L$ as $x \to x_0^+$, then $b(x) \to L$

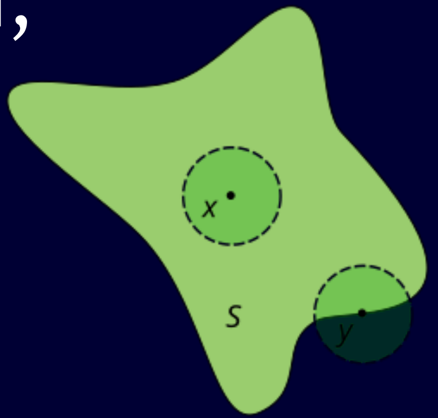
…

- There has to be a better way!

- Analogy: Topology

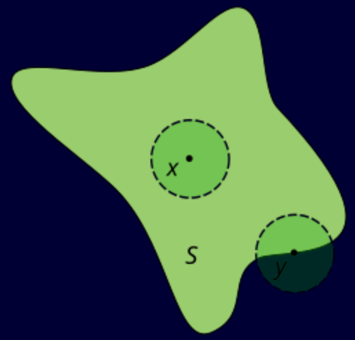
Analysis Lecture 2: “Filters”

- Analogy: Topology
- First: In a metric space, a point is in the “interior” of a set if an ε ball around it is in the set
- And DEF: a set is open if every point in it is an interior point
- Compute: finite intersections of opens are open,
- Arbitrary unions of opens are open
- Empty set, whole space are open



Analysis Lecture 2: “Filters”

- Analogy: Topology
  - First: In a metric space, a point is in the “interior” of a set if an ε ball around it is in the set
  - And DEF: a set is open if every point in it is an interior point



- Better: An Abstract Topology is a collection of sets we declare to be “Open”, including empty set and whole space,
- closed under: finite intersections, and arbitrary unions

Analysis Lecture 2: “Filters”

- Analogy: Topology
- Better: An Abstract Topology is a collection of sets we declare to be “Open”, including empty set and whole space,
- closed under: finite intersections, and arbitrary unions

```
class topologicalSpace (X : Type*) where
  isOpen : Set X → Prop
  isOpen_univ : isOpen (Set.univ : Set X)
  finiteInter : ∀ s t : Set X, isOpen s → isOpen t → isOpen (s ∩ t)
  arbUnion : ∀ (ι : Set (Set X)), (∀ s ∈ ι, isOpen s) → isOpen (⋃ s ∈ ι, s)
```

Analysis Lecture 2: “Filters”

- Analogy: Topology

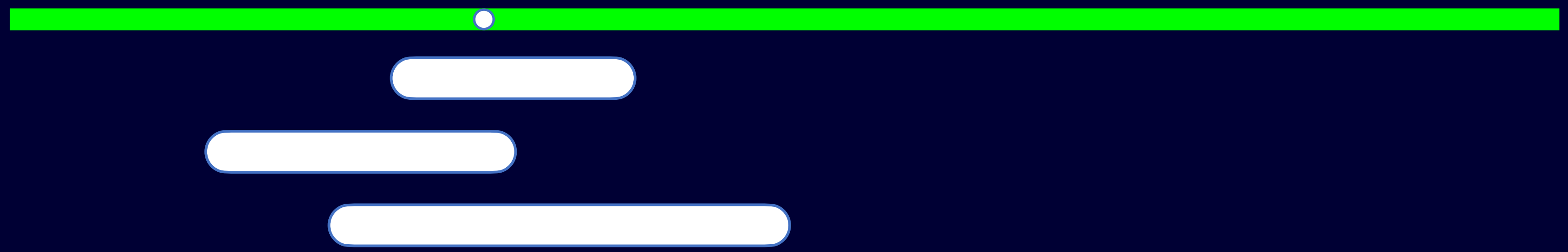
```
class topologicalSpace (X : Type*) where
  isOpen : Set X → Prop
  isOpen_univ : isOpen (Set.univ : Set X)
  finiteInter : ∀ s t : Set X, isOpen s → isOpen t → isOpen (s ∩ t)
  arbUnion : ∀ (ι : Set (Set X)), (∀ s ∈ ι, isOpen s) → isOpen (⋃ s ∈ ι, s)
```

- In Lean, “Set X” is the Type of all functions $X \to$ Prop
- X is not a subset of itself; “univ” is the function X to all 1’s

- How to make the “right” notion of limit?

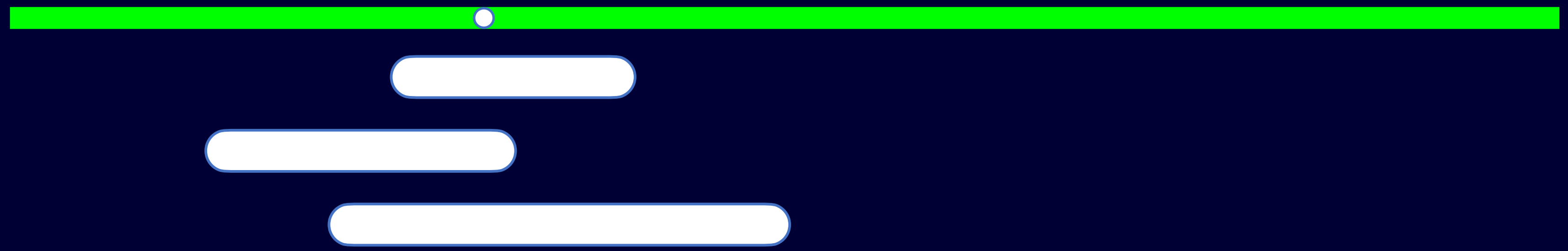
Analysis Lecture 2: “Filters”

- How to make the “right” notion of limit?
- Idea: “zoom in” on $x : X$ along arbitrary open sets.



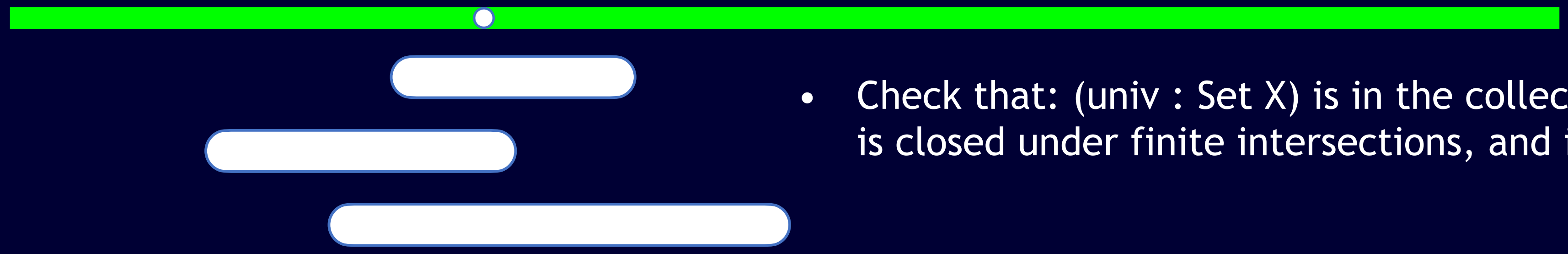
- Def: The “neighborhood filter”, $\mathcal{N}x$, of a point x in a topological space X is: the collection of all Sets of X which contain an open set containing x.

Analysis Lecture 2: "Filters"



- Def: The "neighborhood filter", $\mathcal{N} x$, of a point *x* in a topological space *X* is: the collection of all Sets of *X* which contain an open set containing *x*.
- Check that: (univ : Set X) is in the collection, and collection is closed under finite intersections, and inclusion.
- That's a Filter!

Analysis Lecture 2: “Filters”



- Check that: (univ : Set X) is in the collection, and collection is closed under finite intersections, and inclusion.

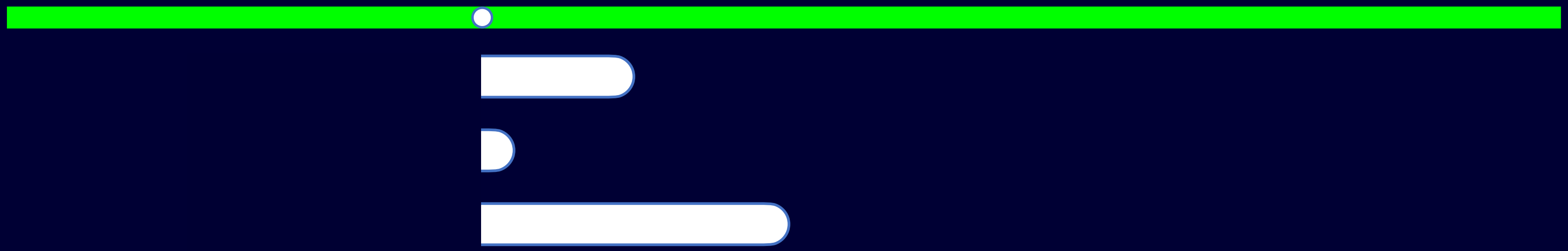
- That’s a Filter!

```
class filter (X : Type*) where
  sets : Set (Set X)
  univ_is : Set.univ ∈ sets
  finiteInter : ∀ s t : Set X, s ∈ sets → t ∈ sets → (s ∩ t) ∈ sets
  inclusion : ∀ s t : Set X, s ∈ sets → s ⊆ t → t ∈ sets
```

Analysis Lecture 2: “Filters”

```
class filter (X : Type*) where
  sets : Set (Set X)
  univ_is : Set.univ ∈ sets
  finiteInter : ∀ s t : Set X, s ∈ sets → t ∈ sets → (s ∩ t) ∈ sets
  inclusion : ∀ s t : Set X, s ∈ sets → s ⊆ t → t ∈ sets
```

- $\mathcal{N}^+ x$



- atTop

Analysis Lecture 2: “Filters”

```
class filter (X : Type*) where
  sets : Set (Set X)
  univ_is : Set.univ ∈ sets
  finiteInter : ∀ s t : Set X, s ∈ sets → t ∈ sets → (s ∩ t) ∈ sets
  inclusion : ∀ s t : Set X, s ∈ sets → s ⊆ t → t ∈ sets
```

- atTop

- cocompact

$(|x| \to \infty)$

Analysis Lecture 2: “Filters”

```
class filter (X : Type*) where
  sets : Set (Set X)
  univ_is : Set.univ ∈ sets
  finiteInter : ∀ s t : Set X, s ∈ sets → t ∈ sets → (s ∩ t) ∈ sets
  inclusion : ∀ s t : Set X, s ∈ sets → s ⊆ t → t ∈ sets
```

- Metric balls are unnecessary for Topology, use abstract “Open”
- Topology is unnecessary for “nhd”, use abstract “Filter”
- Given $f : X \to Y$, how to say that f “TendsTo” y near x?

Analysis Lecture 2: "Filters"

```
class filter (X : Type*) where
  sets : Set (Set X)
  univ_is : Set.univ ∈ sets
  finiteInter : ∀ s t : Set X, s ∈ sets → t ∈ sets → (s ∩ t) ∈ sets
  inclusion : ∀ s t : Set X, s ∈ sets → s ⊆ t → t ∈ sets
```

- Given $f : X \to Y$, how to say that f "TendsTo" y near x?
  - If for all $V \in \mathcal{N}y, f^{-1}V \in \mathcal{N}x$.

```
def tendsto (X Y : Type*) (f : X → Y) (Nhdx : filter X) (Nhdy : filter Y) :
    Prop := ∀ V ∈ Nhdy.sets, f ⁻¹' V ∈ Nhdx.sets
```

- One fell swoop covers sequences, functions, one-sided limits, limits at infinity, etc!

Analysis Lecture 2: “Filters”

```
def tendsto (X Y : Type*) (f : X → Y) (Nhdx : filter X) (Nhdy : filter Y) :
    Prop := ∀ V ∈ Nhdy.sets, f ⁻¹' V ∈ Nhdx.sets
```

- One fell swoop covers sequences, functions, one-sided limits, limits at infinity, etc!
- Squeeze Theorem

```
theorem tendsto_of_tendsto_of_tendsto_of_le_of_le
    {α : Type u} {β : Type v} [TopologicalSpace α]
    [Preorder α] [OrderTopology α] {f g h : β → α}
    {b : Filter β} {a : α} (hg : Filter.Tendsto g b (nhds a))
    (hh : Filter.Tendsto h b (nhds a)) (hgf : g ≤ f)
    (hfh : f ≤ h) :
  Filter.Tendsto f b (nhds a)
```

Analysis Lecture 2: “Filters”

```
def tendsto (X Y : Type*) (f : X → Y) (Nhdx : filter X) (Nhdy : filter Y) :
    Prop := ∀ V ∈ Nhdy.sets, f ⁻¹' V ∈ Nhdx.sets
```

- One fell swoop covers sequences, functions, one-sided limits, limits at infinity, etc!

- Squeeze Theorem

```
theorem tendsto_of_tendsto_of_tendsto_of_le_of_le'
    {α : Type u} {β : Type v} [TopologicalSpace α]
    [Preorder α] [OrderTopology α] {f g h : β → α}
    {b : Filter β} {a : α} (hg : Filter.Tendsto g b (nhds a))
    (hh : Filter.Tendsto h b (nhds a)) (hgf : g ≤ f)
    (hfh : f ≤ h) :
  Filter.Tendsto f b (nhds a)
```

```
(hgf : ∀ᶠ (b : β) in b, g b ≤ f b)
(hfh : ∀ᶠ (b : β) in b, f b ≤ h b) :
```

Analysis Lecture 2: “Filters”

```
def tendsto (X Y : Type*) (f : X → Y) (Nhdx : filter X) (Nhdy : filter Y) :
    Prop := ∀ V ∈ Nhdy.sets, f ⁻¹' V ∈ Nhdx.sets
```

- One fell swoop covers sequences, functions, one-sided limits, limits at infinity, etc!

```
(hgf : ∀ᶠ (b : β) in b, g b ≤ f b)
(hfh : ∀ᶠ (b : β) in b, f b ≤ h b) :
```

- Filter.eventually:

```
/-
A property `p` occurs "eventually" in a filter `f` if the set for which the property holds
is in the filter
-/
def eventually (X : Type*) (p : X → Prop) (f : Filter X) : Prop :=
  { x | p x } ∈ f
```

- Time for exercises!