"By Chain Completeness"...? Proofs on Infinite Lists

Week 2 MT25

Tadayoshi Kamegai

Oxford Compsoc

October 23, 2025

Outline

Motivation

In the Functional Programming course, you will / have learned how to write proofs regarding infinite lists.

Motivation

In the Functional Programming course, you will / have learned how to write proofs regarding infinite lists.
$\rightarrow$ But... much of the background regarding this was skipped.

Motivation

In the Functional Programming course, you will / have learned how to write proofs regarding infinite lists.
$\rightarrow$ But... much of the background regarding this was skipped.

Aim: Give a better background of how the proof works!

Outline

Posets

Definition

A **Poset** or a **Partially ordered set** $(P, \sqsubseteq)$ is a base set P equipped with a binary relation $\sqsubseteq$ that is reflexive, antisymmetric, transitive.

Posets

Definition

A **Poset** or a **Partially ordered set** $(P, \sqsubseteq)$ is a base set P equipped with a binary relation $\sqsubseteq$ that is reflexive, antisymmetric, transitive.

Example

- Any total order is a poset. For example, $(\mathbb{N}, \leq)$, $(\mathbb{Z}, \leq)$, $(\mathbb{R}, \leq)$

Posets

Definition

A **Poset** or a **Partially ordered set** $(P, \sqsubseteq)$ is a base set P equipped with a binary relation $\sqsubseteq$ that is reflexive, antisymmetric, transitive.

Example

- Any total order is a poset. For example, $(\mathbb{N}, \leq)$, $(\mathbb{Z}, \leq)$, $(\mathbb{R}, \leq)$
- Given a base set X, $(\mathcal{P}(X), \subseteq)$, the power set ordered by inclusion

Posets

Definition

A **Poset** or a **Partially ordered set** $(P, \sqsubseteq)$ is a base set P equipped with a binary relation $\sqsubseteq$ that is reflexive, antisymmetric, transitive.

Example

- Any total order is a poset. For example, $(\mathbb{N}, \leq)$, $(\mathbb{Z}, \leq)$, $(\mathbb{R}, \leq)$
- Given a base set X, $(\mathcal{P}(X), \subseteq)$, the power set ordered by inclusion
- $(\mathbb{N}_{\geq 1}, |)$, divisibility order

Posets

Definition

A **Poset** or a **Partially ordered set** $(P, \sqsubseteq)$ is a base set P equipped with a binary relation $\sqsubseteq$ that is reflexive, antisymmetric, transitive.

Example

- Any total order is a poset. For example, $(\mathbb{N}, \leq)$, $(\mathbb{Z}, \leq)$, $(\mathbb{R}, \leq)$
- Given a base set X, $(\mathcal{P}(X), \subseteq)$, the power set ordered by inclusion
- $(\mathbb{N}_{\geq 1}, |)$, divisibility order
- Strings by prefix: $u \preceq v$ if u is a prefix of v

Posets

Definition

A **Poset** or a **Partially ordered set** $(P, \sqsubseteq)$ is a base set P equipped with a binary relation $\sqsubseteq$ that is reflexive, antisymmetric, transitive.

Example

- Any total order is a poset. For example, $(\mathbb{N}, \leq)$, $(\mathbb{Z}, \leq)$, $(\mathbb{R}, \leq)$
- Given a base set X, $(\mathcal{P}(X), \subseteq)$, the power set ordered by inclusion
- $(\mathbb{N}_{\geq 1}, |)$, divisibility order
- Strings by prefix: $u \preceq v$ if u is a prefix of v
- DAGs by reachability

Posets

Definition

A **Poset** or a **Partially ordered set** $(P, \sqsubseteq)$ is a base set P equipped with a binary relation $\sqsubseteq$ that is reflexive, antisymmetric, transitive.

Example

- Any total order is a poset. For example, $(\mathbb{N}, \leq)$, $(\mathbb{Z}, \leq)$, $(\mathbb{R}, \leq)$
- Given a base set X, $(\mathcal{P}(X), \subseteq)$, the power set ordered by inclusion
- $(\mathbb{N}_{\geq 1}, |)$, divisibility order
- Strings by prefix: $u \preceq v$ if u is a prefix of v
- DAGs by reachability
- Abstract interpretation: $a \sqsubseteq b$ if a is less precise than b

Posets

Definition

A **Poset** or a **Partially ordered set** $(P, \sqsubseteq)$ is a base set P equipped with a binary relation $\sqsubseteq$ that is reflexive, antisymmetric, transitive.

Example

- Any total order is a poset. For example, $(\mathbb{N}, \leq)$, $(\mathbb{Z}, \leq)$, $(\mathbb{R}, \leq)$
- Given a base set X, $(\mathcal{P}(X), \subseteq)$, the power set ordered by inclusion
- $(\mathbb{N}_{\geq 1}, |)$, divisibility order
- Strings by prefix: $u \preceq v$ if u is a prefix of v
- DAGs by reachability
- Abstract interpretation: $a \sqsubseteq b$ if a is less precise than b
- For a base set X and a poset D, D^X with $f \leq g \iff \forall x. f(x) \leq g(x)$

Posets

Definition

A **Poset** or a **Partially ordered set** $(P, \sqsubseteq)$ is a base set P equipped with a binary relation $\sqsubseteq$ that is reflexive, antisymmetric, transitive.

Example

- Any total order is a poset. For example, $(\mathbb{N}, \leq)$, $(\mathbb{Z}, \leq)$, $(\mathbb{R}, \leq)$
- Given a base set X, $(\mathcal{P}(X), \subseteq)$, the power set ordered by inclusion
- $(\mathbb{N}_{\geq 1}, |)$, divisibility order
- Strings by prefix: $u \preceq v$ if u is a prefix of v
- DAGs by reachability
- Abstract interpretation: $a \sqsubseteq b$ if a is less precise than b
- For a base set X and a poset D, D^X with $f \leq g \iff \forall x. f(x) \leq g(x)$

Chains

Definition

Given a poset $(P, \sqsubseteq)$, a **chain** C is a subset of P such that every element in C is comparable (totally ordered).

Chains

Definition

Given a poset $(P, \sqsubseteq)$, a **chain** C is a subset of P such that every element in C is comparable (totally ordered).

Definition

The **supremum** of a chain C written $\bigsqcup C$ is the least upper bound (lub) of C in P (if it exists).

Chains

Definition

Given a poset $(P, \sqsubseteq)$, a **chain** C is a subset of P such that every element in C is comparable (totally ordered).

Definition

The **supremum** of a chain C written $\bigsqcup C$ is the least upper bound (lub) of C in P (if it exists).

Definition

The **bottom** element written $\perp$ is the least element of a poset (if it exists).

Chains

Definition

Given a poset $(P, \sqsubseteq)$, a **chain** C is a subset of P such that every element in C is comparable (totally ordered).

Definition

The **supremum** of a chain C written $\bigsqcup C$ is the least upper bound (lub) of C in P (if it exists).

Definition

The **bottom** element written $\perp$ is the least element of a poset (if it exists).

Definition

A **chain-complete poset (ccpo)** is a poset such that every non-empty chain has a least upper bound.

Chains

Definition

Given a poset $(P, \sqsubseteq)$, a **chain** C is a subset of P such that every element in C is comparable (totally ordered).

Definition

The **supremum** of a chain C written $\bigsqcup C$ is the least upper bound (lub) of C in P (if it exists).

Definition

The **bottom** element written $\perp$ is the least element of a poset (if it exists).

Definition

A **chain-complete poset (ccpo)** is a poset such that every non-empty chain has a least upper bound.

$\rightarrow$ Lists form a ccpo with bottom.

Lists as a ccpo

Fix a set A of elements. Consider the poset L of partial lists over A ordered by *information content*:

Lists as a ccpo

Fix a set A of elements. Consider the poset L of partial lists over A ordered by *information content*:

- $\perp$ is the totally undefined list

Lists as a ccpo

Fix a set A of elements. Consider the poset L of partial lists over A ordered by *information content*:

- $\perp$ is the totally undefined list
- A finite list is below any list that has it as a prefix: $x : xs \sqsubseteq y : ys$ iff $x \sqsubseteq y$ and $xs \sqsubseteq ys$

Lists as a ccpo

Fix a set A of elements. Consider the poset L of partial lists over A ordered by *information content*:

- $\perp$ is the totally undefined list
- A finite list is below any list that has it as a prefix: $x : xs \sqsubseteq y : ys$ iff $x \sqsubseteq y$ and $xs \sqsubseteq ys$

For example,

$$\perp \sqsubseteq 0 : \perp \sqsubseteq 0 : 1 : \perp \sqsubseteq 0 : 1 : 2 : \perp \sqsubseteq \cdots$$

Lists as a ccpo

Fix a set A of elements. Consider the poset L of partial lists over A ordered by *information content*:

- $\perp$ is the totally undefined list
- A finite list is below any list that has it as a prefix: $x : xs \sqsubseteq y : ys$ iff $x \sqsubseteq y$ and $xs \sqsubseteq ys$

For example,

$$\perp \sqsubseteq 0 : \perp \sqsubseteq 0 : 1 : \perp \sqsubseteq 0 : 1 : 2 : \perp \sqsubseteq \cdots$$

The $\perp$ acts like an unknown tail, which might terminate or go on forever.

Lists as a ccpo

Fix a set A of elements. Consider the poset L of partial lists over A ordered by *information content*:

- $\perp$ is the totally undefined list
- A finite list is below any list that has it as a prefix: $x : xs \sqsubseteq y : ys$ iff $x \sqsubseteq y$ and $xs \sqsubseteq ys$

For example,

$$\perp \sqsubseteq 0 : \perp \sqsubseteq 0 : 1 : \perp \sqsubseteq 0 : 1 : 2 : \perp \sqsubseteq \cdots$$

The $\perp$ acts like an unknown tail, which might terminate or go on forever. With this chain, the supremum is just [0..].

Lists as a ccpo

Fix a set A of elements. Consider the poset L of partial lists over A ordered by *information content*:

- $\perp$ is the totally undefined list
- A finite list is below any list that has it as a prefix: $x : xs \sqsubseteq y : ys$ iff $x \sqsubseteq y$ and $xs \sqsubseteq ys$

For example,

$$\perp \sqsubseteq 0 : \perp \sqsubseteq 0 : 1 : \perp \sqsubseteq 0 : 1 : 2 : \perp \sqsubseteq \cdots$$

The $\perp$ acts like an unknown tail, which might terminate or go on forever. With this chain, the supremum is just [0..]. But

$$\text{nil} \not\sqsubseteq 0 : \text{nil} \not\sqsubseteq 0 : 1 : \text{nil} \not\sqsubseteq \cdots$$

Lists as a ccpo

Fix a set A of elements. Consider the poset L of partial lists over A ordered by *information content*:

- $\perp$ is the totally undefined list
- A finite list is below any list that has it as a prefix: $x : xs \sqsubseteq y : ys$ iff $x \sqsubseteq y$ and $xs \sqsubseteq ys$

For example,

$$\perp \sqsubseteq 0 : \perp \sqsubseteq 0 : 1 : \perp \sqsubseteq 0 : 1 : 2 : \perp \sqsubseteq \cdots$$

The $\perp$ acts like an unknown tail, which might terminate or go on forever. With this chain, the supremum is just [0..]. But

$$\text{nil} \not\sqsubseteq 0 : \text{nil} \not\sqsubseteq 0 : 1 : \text{nil} \not\sqsubseteq \cdots$$

because finite lists contain the information about termination.

Functions

Definition

Given a poset $(P, \sqsubseteq)$, a function $F : P \to P$ is **monotone** if $x \sqsubseteq y$ implies that $F(x) \sqsubseteq F(y)$

Functions

Definition

Given a poset $(P, \sqsubseteq)$, a function $F : P \to P$ is **monotone** if $x \sqsubseteq y$ implies that $F(x) \sqsubseteq F(y)$

Definition

A function F is **Scott continuous** if it is monotone and preserves least upper bound of chains. That is, given a chain C, we have

$$F(\bigsqcup_{c \in C} c) = \bigsqcup_{c \in C} F(c)$$

Functions

Definition

Given a poset $(P, \sqsubseteq)$, a function $F : P \to P$ is **monotone** if $x \sqsubseteq y$ implies that $F(x) \sqsubseteq F(y)$

Definition

A function F is **Scott continuous** if it is monotone and preserves least upper bound of chains. That is, given a chain C, we have

$$F(\bigsqcup_{c \in C} c) = \bigsqcup_{c \in C} F(c)$$

Aside

This is a continuity based on a topology on posets. In the scott topology, $C \subseteq P$ is closed if

- *C is lower: $y \in C$ and $x \sqsubseteq y$ implies $x \in C$*
- *closed under directed (chain) suprema: when $D \subseteq C$ is directed and $\bigsqcup D$ exists, $\bigsqcup D \in C$*

Functions - continued

Definition

Given a function $F : P \to P$, $x \in P$ is a **fixed point** if $F(x) = x$.

Functions - continued

Definition

Given a function $F : P \to P$, $x \in P$ is a **fixed point** if $F(x) = x$.

Definition

The least fixed point of F, written $\mathrm{lfp}(F)$ is the $\sqsubseteq$-least among fixed points.

Functions - continued

Definition

Given a function $F : P \to P$, $x \in P$ is a **fixed point** if $F(x) = x$.

Definition

The least fixed point of F, written $\mathrm{lfp}(F)$ is the $\sqsubseteq$-least among fixed points.

Theorem (Kleene)

Given a chain complete poset P with bottom and a continuous function $F : P \to P$,

$$\mathrm{lfp}(F) = \bigsqcup_{n \in \mathbb{N}} F^n(\bot)$$

Proofs on Partial Lists

If we consider partial lists that end in $\perp$, there is a clear bijection with lists by sending the bottom element to nil.

Proofs on Partial Lists

If we consider partial lists that end in $\perp$, there is a clear bijection with lists by sending the bottom element to nil. Concretely, setting $S = \{\perp, a_1 : \perp, a_1 : a_2 : \perp, \cdots \mid a_i \in A\}$ we have an order preserving isomorphism between S and $\mathrm{List}_{\mathrm{fin}}(A)$ by

Proofs on Partial Lists

If we consider partial lists that end in $\perp$, there is a clear bijection with lists by sending the bottom element to nil. Concretely, setting $S = \{\perp, a_1 : \perp, a_1 : a_2 : \perp, \cdots \mid a_i \in A\}$ we have an order preserving isomorphism between S and $\mathrm{List}_{\mathrm{fin}}(A)$ by

- $f : \mathrm{List}_{\mathrm{fin}}(A) \to S$ by $f([]) = \perp$, $f(a : xs) = a : f(xs)$
- $g : S \to \mathrm{List}_{\mathrm{fin}}(A)$ by $g(\perp) = []$, $g(a : xs) = a : g(xs)$

Proofs on Partial Lists

If we consider partial lists that end in $\perp$, there is a clear bijection with lists by sending the bottom element to nil. Concretely, setting $S = \{\perp, a_1 : \perp, a_1 : a_2 : \perp, \cdots \mid a_i \in A\}$ we have an order preserving isomorphism between S and $\mathrm{List}_{\mathrm{fin}}(A)$ by

- $f : \mathrm{List}_{\mathrm{fin}}(A) \to S$ by $f([]) = \perp$, $f(a : xs) = a : f(xs)$
- $g : S \to \mathrm{List}_{\mathrm{fin}}(A)$ by $g(\perp) = []$, $g(a : xs) = a : g(xs)$

So, we can prove properties about finite partial lists with a similar induction scheme to finite lists.

Proofs on Partial Lists

If we consider partial lists that end in $\perp$, there is a clear bijection with lists by sending the bottom element to nil. Concretely, setting $S = \{\perp, a_1 : \perp, a_1 : a_2 : \perp, \cdots \mid a_i \in A\}$ we have an order preserving isomorphism between S and $\mathrm{List}_{\mathrm{fin}}(A)$ by

- $f : \mathrm{List}_{\mathrm{fin}}(A) \to S$ by $f([]) = \perp$, $f(a : xs) = a : f(xs)$
- $g : S \to \mathrm{List}_{\mathrm{fin}}(A)$ by $g(\perp) = []$, $g(a : xs) = a : g(xs)$

So, we can prove properties about finite partial lists with a similar induction scheme to finite lists.

Specifically, we just need

- Base: $P(\perp)$
- Step: for all x and xs, $P(xs) \implies P(x : xs)$

Proofs on Partial Lists

If we consider partial lists that end in $\perp$, there is a clear bijection with lists by sending the bottom element to nil. Concretely, setting $S = \{\perp, a_1 : \perp, a_1 : a_2 : \perp, \cdots \mid a_i \in A\}$ we have an order preserving isomorphism between S and $\mathrm{List}_{\mathrm{fin}}(A)$ by

- $f : \mathrm{List}_{\mathrm{fin}}(A) \to S$ by $f([]) = \perp$, $f(a : xs) = a : f(xs)$
- $g : S \to \mathrm{List}_{\mathrm{fin}}(A)$ by $g(\perp) = []$, $g(a : xs) = a : g(xs)$

So, we can prove properties about finite partial lists with a similar induction scheme to finite lists.

Specifically, we just need

- Base: $P(\perp)$
- Step: for all x and xs, $P(xs) \implies P(x : xs)$

Then, for all $xs \in S$, $P(xs)$.

Proofs on Partial Lists

If we consider partial lists that end in $\perp$, there is a clear bijection with lists by sending the bottom element to nil. Concretely, setting $S = \{\perp, a_1 : \perp, a_1 : a_2 : \perp, \cdots \mid a_i \in A\}$ we have an order preserving isomorphism between S and $\mathrm{List}_{\mathrm{fin}}(A)$ by

- $f : \mathrm{List}_{\mathrm{fin}}(A) \to S$ by $f([]) = \perp$, $f(a : xs) = a : f(xs)$
- $g : S \to \mathrm{List}_{\mathrm{fin}}(A)$ by $g(\perp) = []$, $g(a : xs) = a : g(xs)$

So, we can prove properties about finite partial lists with a similar induction scheme to finite lists.

Specifically, we just need

- Base: $P(\perp)$
- Step: for all x and xs, $P(xs) \implies P(x : xs)$

Then, for all $xs \in S$, $P(xs)$. However, this doesn't give any properties about total infinite lists, as they live outside of S.

Proofs on Partial Lists

If we consider partial lists that end in $\perp$, there is a clear bijection with lists by sending the bottom element to nil. Concretely, setting $S = \{\perp, a_1 : \perp, a_1 : a_2 : \perp, \cdots \mid a_i \in A\}$ we have an order preserving isomorphism between S and $\mathrm{List}_{\mathrm{fin}}(A)$ by

- $f : \mathrm{List}_{\mathrm{fin}}(A) \to S$ by $f([]) = \perp$, $f(a : xs) = a : f(xs)$
- $g : S \to \mathrm{List}_{\mathrm{fin}}(A)$ by $g(\perp) = []$, $g(a : xs) = a : g(xs)$

So, we can prove properties about finite partial lists with a similar induction scheme to finite lists.

Specifically, we just need

- Base: $P(\perp)$
- Step: for all x and xs, $P(xs) \implies P(x : xs)$

Then, for all $xs \in S$, $P(xs)$. However, this doesn't give any properties about total infinite lists, as they live outside of S.
To do this, we introduce the notion of admissible predicates.

Admissibility

Definition

A predicate $P : L \to \{\text{true}, \text{false}\}$ is **admissible** if it is closed under least upper bounds of chains. That is, if

- $x_0 \sqsubseteq x_1 \sqsubseteq \cdots$
- for all n, $P(x_n)$

implies that $P(\bigsqcup_n x_n)$

Admissibility

Definition

A predicate $P : L \to \{\text{true}, \text{false}\}$ is **admissible** if it is closed under least upper bounds of chains. That is, if

- $x_0 \sqsubseteq x_1 \sqsubseteq \cdots$
- for all n, $P(x_n)$

implies that $P(\bigsqcup_n x_n)$

Intuition: if every finite approximation satisfies P, then the limit (possibly infinite) also satisfies P.

Proofs on Partial Lists - Continued

We give a method to prove propositions on infinite lists.

Proofs on Partial Lists - Continued

We give a method to prove propositions on infinite lists.

Suppose that P is an admissible predicate.

Proofs on Partial Lists - Continued

We give a method to prove propositions on infinite lists.

Suppose that P is an admissible predicate. Let xs be an infinite list, and write

$$xs := x_0 : x_1 : x_2 : \cdots$$

Proofs on Partial Lists - Continued

We give a method to prove propositions on infinite lists.

Suppose that P is an admissible predicate. Let xs be an infinite list, and write

$$xs := x_0 : x_1 : x_2 : \cdots$$

Then define a chain C by,

$$\perp \sqsubseteq x_0 : \perp \sqsubseteq x_0 : x_1 : \perp \sqsubseteq \cdots$$

Proofs on Partial Lists - Continued

We give a method to prove propositions on infinite lists.

Suppose that P is an admissible predicate. Let xs be an infinite list, and write

$$xs := x_0 : x_1 : x_2 : \cdots$$

Then define a chain C by,

$$\perp \sqsubseteq x_0 : \perp \sqsubseteq x_0 : x_1 : \perp \sqsubseteq \cdots$$

By construction, $\bigsqcup C = x_0 : x_1 : \cdots = xs$

Proofs on Partial Lists - Continued

We give a method to prove propositions on infinite lists.

Suppose that P is an admissible predicate. Let xs be an infinite list, and write

$$xs := x_0 : x_1 : x_2 : \cdots$$

Then define a chain C by,

$$\perp \sqsubseteq x_0 : \perp \sqsubseteq x_0 : x_1 : \perp \sqsubseteq \cdots$$

By construction, $\bigsqcup C = x_0 : x_1 : \cdots = xs$

As P is admissible, to prove a predicate about xs, it suffices to prove it is the case for every element in C.

Proofs on Partial Lists - Continued

We give a method to prove propositions on infinite lists.

Suppose that P is an admissible predicate. Let xs be an infinite list, and write

$$xs := x_0 : x_1 : x_2 : \cdots$$

Then define a chain C by,

$$\perp \sqsubseteq x_0 : \perp \sqsubseteq x_0 : x_1 : \perp \sqsubseteq \cdots$$

By construction, $\bigsqcup C = x_0 : x_1 : \cdots = xs$

As P is admissible, to prove a predicate about xs, it suffices to prove it is the case for every element in C. As elements in C are finite partial lists, we can use the proof scheme from before.

Outline

Did we just shift our problem?

Although we have a scheme to prove properties about infinite lists now, we still need to show that P is admissible.

Did we just shift our problem?

Although we have a scheme to prove properties about infinite lists now, we still need to show that P is admissible.
$\rightarrow$ Feels like we just moved the problem backwards.

Did we just shift our problem?

Although we have a scheme to prove properties about infinite lists now, we still need to show that P is admissible.
$\rightarrow$ Feels like we just moved the problem backwards.

Some questions remain. . .

Did we just shift our problem?

Although we have a scheme to prove properties about infinite lists now, we still need to show that P is admissible.
$\rightarrow$ Feels like we just moved the problem backwards.

Some questions remain. . .

- What do admissible predicates look like?

Did we just shift our problem?

Although we have a scheme to prove properties about infinite lists now, we still need to show that P is admissible.
$\rightarrow$ Feels like we just moved the problem backwards.

Some questions remain. . .

- What do admissible predicates look like?
- Is the proposition I want to prove admissible?

What does it mean to be admissible?

Recall,

Definition

A predicate $P : L \to \{\text{true}, \text{false}\}$ is **admissible** if

- for any chain $x_0 \sqsubseteq x_1 \sqsubseteq \cdots$
- if we can show for all n, $P(x_n)$

implies that $P(\bigsqcup_n x_n)$

What does it mean to be admissible?

Recall,

Definition

A predicate $P : L \to \{\text{true}, \text{false}\}$ is **admissible** if

- for any chain $x_0 \sqsubseteq x_1 \sqsubseteq \cdots$
- if we can show for all n, $P(x_n)$

implies that $P(\bigsqcup_n x_n)$

Some properties about lists aren't admissible.

What does it mean to be admissible?

Recall,

Definition

A predicate $P : L \to \{\text{true}, \text{false}\}$ is **admissible** if

- for any chain $x_0 \sqsubseteq x_1 \sqsubseteq \cdots$
- if we can show for all n, $P(x_n)$

implies that $P(\bigsqcup_n x_n)$

Some properties about lists aren't admissible. For example,

- xs is finite

What does it mean to be admissible?

Recall,

Definition

A predicate $P : L \to \{\text{true}, \text{false}\}$ is **admissible** if

- for any chain $x_0 \sqsubseteq x_1 \sqsubseteq \cdots$
- if we can show for all n, $P(x_n)$

implies that $P(\bigsqcup_n x_n)$

Some properties about lists aren't admissible. For example,

- xs is finite
- $\exists n.\text{drop } n \; xs = \bot$

What does it mean to be admissible?

Recall,

Definition

A predicate $P : L \to \{\text{true}, \text{false}\}$ is **admissible** if

- for any chain $x_0 \sqsubseteq x_1 \sqsubseteq \cdots$
- if we can show for all n, $P(x_n)$

implies that $P(\bigsqcup_n x_n)$

Some properties about lists aren't admissible. For example,

- xs is finite
- $\exists n.\text{drop } n\ xs = \bot$

These are examples of 'limit-fragile' propositions.

Safety Properties are Admissible

A safety property say that 'bad things never happen'.

Safety Properties are Admissible

A safety property say that 'bad things never happen'. A property is **safe** if having all finite prefixes of a list x lie in P implies that $x \in P$.

Safety Properties are Admissible

A safety property say that 'bad things never happen'. A property is **safe** if having all finite prefixes of a list x lie in P implies that $x \in P$. Equivalently, if $x \notin P$, there exists a finite prefix $y \sqsubseteq x$ with $y \notin P$ (a finite counterexample).

Safety Properties are Admissible

A safety property say that 'bad things never happen'. A property is **safe** if having all finite prefixes of a list x lie in P implies that $x \in P$. Equivalently, if $x \notin P$, there exists a finite prefix $y \sqsubseteq x$ with $y \notin P$ (a finite counterexample).
There are many examples of safety properties on lists.

Safety Properties are Admissible

A safety property say that 'bad things never happen'. A property is **safe** if having all finite prefixes of a list x lie in P implies that $x \in P$. Equivalently, if $x \notin P$, there exists a finite prefix $y \sqsubseteq x$ with $y \notin P$ (a finite counterexample).
There are many examples of safety properties on lists.
For example, where we ban certain patterns:

Safety Properties are Admissible

A safety property say that 'bad things never happen'. A property is **safe** if having all finite prefixes of a list x lie in P implies that $x \in P$. Equivalently, if $x \notin P$, there exists a finite prefix $y \sqsubseteq x$ with $y \notin P$ (a finite counterexample).
There are many examples of safety properties on lists.
For example, where we ban certain patterns:

- No 1 ever occurs

Safety Properties are Admissible

A safety property say that 'bad things never happen'. A property is **safe** if having all finite prefixes of a list x lie in P implies that $x \in P$. Equivalently, if $x \notin P$, there exists a finite prefix $y \sqsubseteq x$ with $y \notin P$ (a finite counterexample).
There are many examples of safety properties on lists.
For example, where we ban certain patterns:

- No 1 ever occurs
- No two consecutive 1s

Safety Properties are Admissible

A safety property say that ‘bad things never happen’. A property is **safe** if having all finite prefixes of a list x lie in P implies that $x \in P$. Equivalently, if $x \notin P$, there exists a finite prefix $y \sqsubseteq x$ with $y \notin P$ (a finite counterexample).
There are many examples of safety properties on lists.
For example, where we ban certain patterns:

- No 1 ever occurs
- No two consecutive 1s
- All elements are less than 10

Safety Properties are Admissible

A safety property say that 'bad things never happen'. A property is **safe** if having all finite prefixes of a list x lie in P implies that $x \in P$. Equivalently, if $x \notin P$, there exists a finite prefix $y \sqsubseteq x$ with $y \notin P$ (a finite counterexample).
There are many examples of safety properties on lists.
For example, where we ban certain patterns:

- No 1 ever occurs
- No two consecutive 1s
- All elements are less than 10
- We never see "010"

Safety Properties are Admissible

A safety property say that ‘bad things never happen’. A property is **safe** if having all finite prefixes of a list x lie in P implies that $x \in P$. Equivalently, if $x \notin P$, there exists a finite prefix $y \sqsubseteq x$ with $y \notin P$ (a finite counterexample).
There are many examples of safety properties on lists.
For example, where we ban certain patterns:

- No 1 ever occurs
- No two consecutive 1s
- All elements are less than 10
- We never see "010"

Or when the proposition is prefix-invariant (bad patterns can be checked with a finite prefix):

Safety Properties are Admissible

A safety property say that ‘bad things never happen’. A property is **safe** if having all finite prefixes of a list x lie in P implies that $x \in P$. Equivalently, if $x \notin P$, there exists a finite prefix $y \sqsubseteq x$ with $y \notin P$ (a finite counterexample).
There are many examples of safety properties on lists.
For example, where we ban certain patterns:

- No 1 ever occurs
- No two consecutive 1s
- All elements are less than 10
- We never see "010"

Or when the proposition is prefix-invariant (bad patterns can be checked with a finite prefix):

- At most 1 in any prefix

Safety Properties are Admissible

A safety property say that 'bad things never happen'. A property is **safe** if having all finite prefixes of a list x lie in P implies that $x \in P$. Equivalently, if $x \notin P$, there exists a finite prefix $y \sqsubseteq x$ with $y \notin P$ (a finite counterexample).
There are many examples of safety properties on lists.
For example, where we ban certain patterns:

- No 1 ever occurs
- No two consecutive 1s
- All elements are less than 10
- We never see "010"

Or when the proposition is prefix-invariant (bad patterns can be checked with a finite prefix):

- At most 1 in any prefix
- Nondecreasing list of numbers

Safety Properties are Admissible

A safety property say that 'bad things never happen'. A property is **safe** if having all finite prefixes of a list x lie in P implies that $x \in P$. Equivalently, if $x \notin P$, there exists a finite prefix $y \sqsubseteq x$ with $y \notin P$ (a finite counterexample).
There are many examples of safety properties on lists.
For example, where we ban certain patterns:

- No 1 ever occurs
- No two consecutive 1s
- All elements are less than 10
- We never see "010"

Or when the proposition is prefix-invariant (bad patterns can be checked with a finite prefix):

- At most 1 in any prefix
- Nondecreasing list of numbers
- Every 1 is immediately followed by a 0

Scott-closed sets give admissible predicates

Recall,

Definition

$C \subseteq P$ is scott-closed if

- C is lower: $y \in C$ and $x \sqsubseteq y$ implies $x \in C$
- closed under directed (chain) suprema: when $D \subseteq C$ is directed and $\bigsqcup D$ exists, $\bigsqcup D \in C$

Scott-closed sets give admissible predicates

Recall,

Definition

$C \subseteq P$ is scott-closed if

- C is lower: $y \in C$ and $x \sqsubseteq y$ implies $x \in C$
- closed under directed (chain) suprema: when $D \subseteq C$ is directed and $\bigsqcup D$ exists, $\bigsqcup D \in C$

Let $C \subseteq E$ be Scott-closed and $f : D \to E$ be a Scott-continuous function.

Scott-closed sets give admissible predicates

Recall,

Definition

$C \subseteq P$ is scott-closed if

- C is lower: $y \in C$ and $x \sqsubseteq y$ implies $x \in C$
- closed under directed (chain) suprema: when $D \subseteq C$ is directed and $\bigsqcup D$ exists, $\bigsqcup D \in C$

Let $C \subseteq E$ be Scott-closed and $f : D \to E$ be a Scott-continuous function. If we define a property P by $P(x) \iff f(x) \in C$ then by continuity, the preimage $f^{-1}(C) = \{x \mid f(x) \in C\} = \{x \mid P(x)\}$ is Scott-closed.

Scott-closed sets give admissible predicates

Recall,

Definition

$C \subseteq P$ is scott-closed if

- C is lower: $y \in C$ and $x \sqsubseteq y$ implies $x \in C$
- closed under directed (chain) suprema: when $D \subseteq C$ is directed and $\bigsqcup D$ exists, $\bigsqcup D \in C$

Let $C \subseteq E$ be Scott-closed and $f : D \to E$ be a Scott-continuous function. If we define a property P by $P(x) \iff f(x) \in C$ then by continuity, the preimage $f^{-1}(C) = \{x \mid f(x) \in C\} = \{x \mid P(x)\}$ is Scott-closed. Hence, to show P is admissible, it suffices to realize P as a preimage of a closed set along a Scott-continuous map.

Scott-closed sets give admissible predicates

Recall,

Definition

$C \subseteq P$ is scott-closed if

- C is lower: $y \in C$ and $x \sqsubseteq y$ implies $x \in C$
- closed under directed (chain) suprema: when $D \subseteq C$ is directed and $\bigsqcup D$ exists, $\bigsqcup D \in C$

Let $C \subseteq E$ be Scott-closed and $f : D \to E$ be a Scott-continuous function. If we define a property P by $P(x) \iff f(x) \in C$ then by continuity, the preimage $f^{-1}(C) = \{x \mid f(x) \in C\} = \{x \mid P(x)\}$ is Scott-closed. Hence, to show P is admissible, it suffices to realize P as a preimage of a closed set along a Scott-continuous map.

To then find continuous maps f, we note that

- Composition of continuous constructors
- Composition of continuous folds
- Products
- Evaluation of definable expressions

are all Scott-continuous.

Positive properties are built to be closed

- When we say ‘positive’, we mean propositions built using continuous things with $\forall$ and $\wedge$ but no $\exists$ or $\vee$.

Positive properties are built to be closed

- When we say 'positive', we mean propositions built using continuous things with $\forall$ and $\wedge$ but no $\exists$ or $\vee$.
- Universal quantification and conjunctions correspond to intersections of Scott-closed sets, and the property of closedness is preserved under arbitrary intersection.

Positive properties are built to be closed

- When we say 'positive', we mean propositions built using continuous things with $\forall$ and $\wedge$ but no $\exists$ or $\vee$.
- Universal quantification and conjunctions correspond to intersections of Scott-closed sets, and the property of closedness is preserved under arbitrary intersection.
- For example, $\forall i.P_i$ has truth set $\bigcap_i C_{P_i}$, and this remains closed.

Positive properties are built to be closed

- When we say ‘positive’, we mean propositions built using continuous things with $\forall$ and $\wedge$ but no $\exists$ or $\vee$.
- Universal quantification and conjunctions correspond to intersections of Scott-closed sets, and the property of closedness is preserved under arbitrary intersection.
- For example, $\forall i.P_i$ has truth set $\bigcap_i C_{P_i}$, and this remains closed.
- $\exists i.P_i$ corresponds to $\bigcup_i C_{P_i}$, but unions of Scott-closed sets need not be Scott-closed.

Positive properties are built to be closed

- When we say 'positive', we mean propositions built using continuous things with $\forall$ and $\wedge$ but no $\exists$ or $\vee$.
- Universal quantification and conjunctions correspond to intersections of Scott-closed sets, and the property of closedness is preserved under arbitrary intersection.
- For example, $\forall i.P_i$ has truth set $\bigcap_i C_{P_i}$, and this remains closed.

- $\exists i.P_i$ corresponds to $\bigcup_i C_{P_i}$, but unions of Scott-closed sets need not be Scott-closed.
- When we have $P(xs) \iff \exists n.\text{drop } n \; xs = \perp$, the basic disjunct is closed, but the union is not closed under limits.

Our general picture

- Lists form a ccpo with a bottom element.

Our general picture

- Lists form a ccpo with a bottom element.
- Given an admissible predicate, we can prove properties about infinite lists.

Our general picture

- Lists form a ccpo with a bottom element.
- Given an admissible predicate, we can prove properties about infinite lists.
- Many properties about lists are indeed admissible.

Our general picture

- Lists form a ccpo with a bottom element.
- Given an admissible predicate, we can prove properties about infinite lists.
- Many properties about lists are indeed admissible.

Further. . .

Our general picture

- Lists form a ccpo with a bottom element.
- Given an admissible predicate, we can prove properties about infinite lists.
- Many properties about lists are indeed admissible.

Further. . .

- We can generalize this to other algebraic objects, not just lists (and we can generate the inductive scheme given a suitable functor that describes it)

Questions?