# Logical Predicates

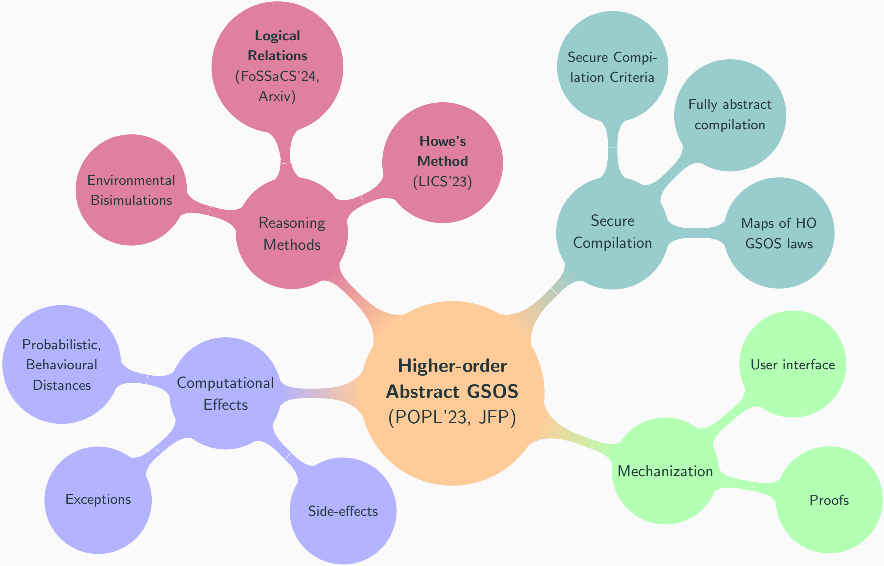
in Higher-order Mathematical Operational Semantics

Sergey Goncharov, Alessio Santamaria, Lutz Schröder, **Stelios Tsampas** and Henning Urbat

FoSSaCS 2024

Friedrich-Alexander-Universität Erlangen-Nürnberg

# Higher-Order Mathematical Operational Semantics (or HO Abstract GSOS)



## The setting of Logical Predicates

1. An operational semantics of a higher-order language
   - Typically a typed $\lambda$-calculus.
   - Write $\Lambda_\tau(\Gamma)$ for the set $\{t \mid \Gamma \vdash t\colon \tau\}$ and $\Lambda_\tau$ for the set $\{t \mid \varnothing \vdash t\colon \tau\}$.

## The setting of Logical Predicates

1. An operational semantics of a higher-order language
   - Typically a typed $\lambda$-calculus.
   - Write $\Lambda_\tau(\Gamma)$ for the set $\{t \mid \Gamma \vdash t \colon \tau\}$ and $\Lambda_\tau$ for the set $\{t \mid \varnothing \vdash t \colon \tau\}$.
2. A (type-indexed) predicate $P \rightarrowtail \Lambda$, that can't be proven inductively
   - Family $(P_\tau \subseteq \Lambda_\tau)_{\tau \in \mathsf{Ty}}$
   - Strong normalization, type safety etc.

# The setting of Logical Predicates

1. An operational semantics of a higher-order language
    - Typically a typed $\lambda$-calculus.
    - Write $\Lambda_\tau(\Gamma)$ for the set $\{t \mid \Gamma \vdash t \colon \tau\}$ and $\Lambda_\tau$ for the set $\{t \mid \varnothing \vdash t \colon \tau\}$.
2. A (type-indexed) predicate $P \rightarrowtail \Lambda$, that can't be proven inductively
    - Family $(P_\tau \subseteq \Lambda_\tau)_{\tau \in \mathsf{Ty}}$
    - Strong normalization, type safety etc.
3. We construct a suitable *logical predicate over* $P$, say $\Box P$, which implies $P$.
    - Logical in the sense that

      "*For any term $t$ and $s$ in $\Box P$ and of the suitable type, $t \cdot s$ is also in $\Box P$*".

## The setting of Logical Predicates

1. An operational semantics of a higher-order language
    - Typically a typed $\lambda$-calculus.
    - Write $\Lambda_\tau(\Gamma)$ for the set $\{t \mid \Gamma \vdash t\colon \tau\}$ and $\Lambda_\tau$ for the set $\{t \mid \varnothing \vdash t\colon \tau\}$.
2. A (type-indexed) predicate $P \rightarrowtail \Lambda$, that can't be proven inductively
    - Family $(P_\tau \subseteq \Lambda_\tau)_{\tau \in \mathsf{Ty}}$
    - Strong normalization, type safety etc.
3. We construct a suitable *logical predicate over* $P$, say $\Box P$, which implies $P$.
    - Logical in the sense that

      "*For any term t and s in* $\Box P$ *and of the suitable type,* $t \cdot s$ *is also in* $\Box P$".
4. Proceed by induction to prove that (the open extension of) $\Box P$ holds.

# Strong Normalization

## Definition (A standard logical predicate)

$$\mathrm{SN}_{\mathsf{unit}}\,(t) = \Downarrow_{\mathsf{unit}}\,(t)$$
$$\mathrm{SN}_{\tau_1 \to \tau_2}\,(t) = \Downarrow_{\tau_1 \to \tau_2}\,(t) \wedge (\forall s \colon \tau_1 .\, \mathrm{SN}_{\tau_1}(s) \implies \mathrm{SN}_{\tau_2}(t \cdot s))$$

## Strong Normalization

**Definition (A standard logical predicate)**

$$\mathrm{SN}_{\mathsf{unit}}(t) = \Downarrow_{\mathsf{unit}}(t)$$
$$\mathrm{SN}_{\tau_1 \rightarrow \tau_2}(t) = \Downarrow_{\tau_1 \rightarrow \tau_2}(t) \wedge (\forall s : \tau_1 .\, \mathrm{SN}_{\tau_1}(s) \implies \mathrm{SN}_{\tau_2}(t \cdot s))$$

**Definition (Open extension of $\mathrm{SN}$)**

$$\vec{\mathrm{SN}}_{\tau}(t)(\Gamma) = \text{For any closed substitution } (\varnothing \vdash e_n : \Gamma(n))_{n \in |\Gamma|}$$
$$\text{such that } \forall n \in |\Gamma| .\, \mathrm{SN}_{\Gamma(n)}(e_n), \text{ then } \mathrm{SN}_{\tau}(t[e_n / x_n])$$

# Strong Normalization

One annoying case of the proof is that of $\lambda$-abstraction $\Gamma \vdash \lambda x\colon \tau_1.t\colon \tau_1 \rightarrow \tau_2$.
Given a substitution $(\varnothing \vdash e_n\colon \Gamma(n))_{n\in|\Gamma|}$ satisftying $\mathrm{SN}$, we have to:

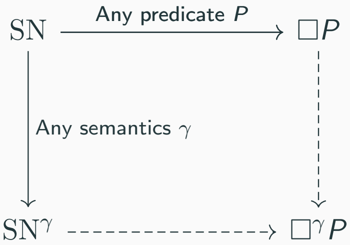
- Push the substitution inside the $\lambda$-abstraction, try to prove that the whole term is in $\mathrm{SN}$, for that reason consider what happens when we have terms such as $(\lambda x\colon \tau_1.t') \cdot s$ with $\mathrm{SN}_{\tau_1}(s)$ for the substituted $t'$, think back to what happens during $\beta$-reduction, reflect on properties of substitution etc.

Complex language $\implies$ complex argument...

# The goal of this talk

I will argue for two directions of abstraction, via

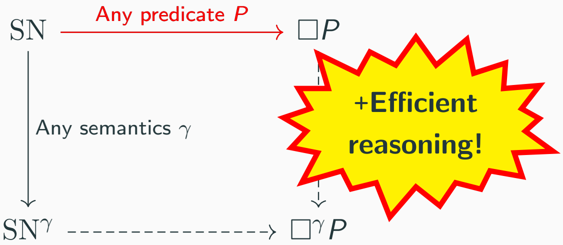
**Higher-order Abstract GSOS**



# The goal of this talk

I will argue for two directions of abstraction, via

**Higher-order Abstract GSOS**



# Dissecting the logical predicate (1)

$$\mathrm{SN}_{\mathsf{unit}}(t) = \Downarrow_{\mathsf{unit}}(t)$$

$$\mathrm{SN}_{\tau_1 \rightarrow \tau_2}(t) = \Downarrow_{\tau_1 \rightarrow \tau_2}(t) \wedge (\forall s \colon \tau_1 .\, \mathrm{SN}_{\tau_1}(s) \implies \mathrm{SN}_{\tau_2}(t \cdot s))$$

# Dissecting the logical predicate (1)

$$\mathrm{SN}_{\mathsf{unit}}(t) = \Downarrow_{\mathsf{unit}}(t)$$
$$\mathrm{SN}_{\tau_1 \to \tau_2}(t) = \Downarrow_{\tau_1 \to \tau_2}(t) \wedge (\forall s \colon \tau_1.\, \mathrm{SN}_{\tau_1}(s) \implies \mathrm{SN}_{\tau_2}(t \cdot s))$$

Idea : Write $t \overset{s}{\Rightarrow} t'$ if $t \Downarrow \lambda x \colon \tau_1.M$ and $t' = M[s/x]$

# Dissecting the logical predicate (1)

$$\mathrm{SN}_{\mathsf{unit}}(t) = \Downarrow_{\mathsf{unit}}(t)$$
$$\mathrm{SN}_{\tau_1 \to \tau_2}(t) = \Downarrow_{\tau_1 \to \tau_2}(t) \wedge (\forall s\colon \tau_1.\, \mathrm{SN}_{\tau_1}(s) \implies \mathrm{SN}_{\tau_2}(t \cdot s))$$

Idea : Write $t \overset{s}{\Rightarrow} t'$ if $t \Downarrow \lambda x\colon \tau_1.M$ and $t' = M[s/x]$

$$\Downarrow\!\!\Downarrow_{\mathsf{unit}}(t) = \Downarrow_{\mathsf{unit}}(t)$$
$$\Downarrow\!\!\Downarrow_{\tau_1 \to \tau_2}(t) = \Downarrow_{\tau_1 \to \tau_2} t \wedge (\forall s\colon \tau_1.\, t \overset{s}{\Rightarrow} t' \wedge \Downarrow\!\!\Downarrow_{\tau_1}(s) \implies \Downarrow\!\!\Downarrow_{\tau_2}(t'))$$

# Dissecting the logical predicate (1)

$$\mathrm{SN}_{\mathsf{unit}}(t) = \Downarrow_{\mathsf{unit}}(t)$$
$$\mathrm{SN}_{\tau_1 \to \tau_2}(t) = \Downarrow_{\tau_1 \to \tau_2}(t) \wedge (\forall s\colon \tau_1.\, \mathrm{SN}_{\tau_1}(s) \implies \mathrm{SN}_{\tau_2}(t \cdot s))$$

Idea : Write $t \stackrel{s}{\Rightarrow} t'$ if $t \Downarrow \lambda x\colon \tau_1.M$ and $t' = M[s/x]$

$$\Downarrow\!\!\Downarrow_{\mathsf{unit}}(t) = \Downarrow_{\mathsf{unit}}(t)$$
$$\Downarrow\!\!\Downarrow_{\tau_1 \to \tau_2}(t) = \Downarrow_{\tau_1 \to \tau_2} t \wedge (\forall s\colon \tau_1.\, t \stackrel{s}{\Rightarrow} t' \wedge \Downarrow\!\!\Downarrow_{\tau_1}(s) \implies \Downarrow\!\!\Downarrow_{\tau_2}(t'))$$

Idea : Abstract away from the predicate $\Downarrow$

# Dissecting the logical predicate (2)

$$\Box P_{\text{unit}}\,(t) = P_{\text{unit}}\,(t)$$

$$\Box P_{\tau_1 \twoheadrightarrow \tau_2}\,(t) = P_{\tau_1 \twoheadrightarrow \tau_2}\, t \land (\forall s \colon \tau_1.\, t \overset{s}{\Rightarrow} t' \land \Box P_{\tau_1}\,(s) \implies \Box P_{\tau_2}\,(t'))$$

# Dissecting the logical predicate (2)

$$\Box P_{\text{unit}}(t) = P_{\text{unit}}(t)$$
$$\Box P_{\tau_1 \rightarrowtriangle \tau_2}(t) = P_{\tau_1 \rightarrowtriangle \tau_2}\, t \wedge (\forall s \colon \tau_1.\, t \overset{s}{\Rightarrow} t' \wedge \Box P_{\tau_1}(s) \implies \Box P_{\tau_2}(t'))$$

Idea : Move one from $\Rightarrow$ to the more fundamental $\rightarrow$

# Dissecting the logical predicate (2)

$$\Box P_{\text{unit}}(t) = P_{\text{unit}}(t)$$

$$\Box P_{\tau_1 \to\!\triangleright \tau_2}(t) = P_{\tau_1 \to\!\triangleright \tau_2}\, t \wedge (\forall s\colon \tau_1.\, t \overset{s}{\Rightarrow} t' \wedge \Box P_{\tau_1}(s) \implies \Box P_{\tau_2}(t'))$$

Idea : Move one from $\Rightarrow$ to the more fundamental $\rightarrow$

greatest subset of $\Lambda_{\tau_1 \to\!\triangleright \tau_2}$

$$\Box P_{\text{unit}}(t) = P_{\text{unit}}(t)$$

$$\Box P_{\tau_1 \to\!\triangleright \tau_2}(t) \implies P_{\tau_1 \to\!\triangleright \tau_2}(t) \wedge \begin{cases} \Box P_{\tau_1 \to\!\triangleright \tau_2}(t') & \text{if} \quad t \rightarrow t' \\ \Box P_{\tau_1}(s) \implies \Box P_{\tau_2}(t') & \text{if} \quad t \overset{s}{\rightarrow} t' \end{cases}$$

**Theorem**

*Let $P \rightarrowtail \Lambda$ be any predicate on closed terms. Then $P$ is true if all of the following are true:*

1. *the unit expression* $\mathrm{e} : \mathrm{unit}$ *satisfies* ~~$\Box_{\mathrm{unit}} P$~~ $P_{\mathrm{unit}}$,
2. *for all closed application terms* $t\,s$ *such that* $\Box_{\tau_1 \to \tau_2} P(t)$ *and* $\Box_{\tau_1} P(s)$, *we have* ~~$\Box_{\tau_2} P(t\,s)$~~ $P_{\tau_2}(t\,s)$, *and*
3. *for all* $\lambda$*-abstractions* $\lambda x : \tau_1 .\, t : \tau_1 \to \tau_2$, *such that* $\lambda x : \tau_1 .\, t$ *is in the open extension of* $\Box P$ *and given a substitution* $\vec{e}$ *that satisfies* $\Box P$, $(\lambda x : \tau_1 .\, t)[\vec{e}/\vec{x}]$, *we have that* $(\lambda x : \tau_1 .\, t)[\vec{e}/\vec{x}]$ *is in* ~~$\Box P$~~, $P$.

**Proof.**

Instantiate Th. 36 with $(\mathrm{Th36}.P)_\tau(\varnothing) = P_\tau$ and $(\mathrm{Th36}.P)_\tau(\Gamma \neq \varnothing) = \top$. □

# Let's try this out!

## Proving strong normalization for STLC

1. $\Downarrow_{\mathsf{unit}} (\mathsf{e})$;
2. $\Downarrow_{\tau_2} (t\, s)$ with $\Box_{\tau_1 \rightarrow \tau_2} \Downarrow (t)$ and $\Box_{\tau_1} \Downarrow (s)$;
3. $\Downarrow_{\tau_1 \rightarrow \tau_2} (\lambda x \colon \tau_1 .\, t)$ (what $t$ can do is irrelevant in this case).

# Let's try this out!

## Proving strong normalization for STLC

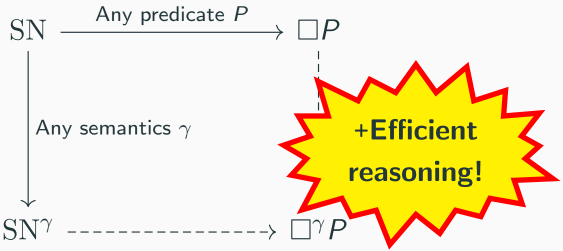
1. $\Downarrow_{\text{unit}}$ (e);
2. $\Downarrow_{\tau_2} (t\, s)$ with $\Box_{\tau_1 \twoheadrightarrow \tau_2} \Downarrow (t)$ and $\Box_{\tau_1} \Downarrow (s)$;
3. $\Downarrow_{\tau_1 \twoheadrightarrow \tau_2} (\lambda x\colon \tau_1.\, t)$ (what $t$ can do is irrelevant in this case).

## Proof.

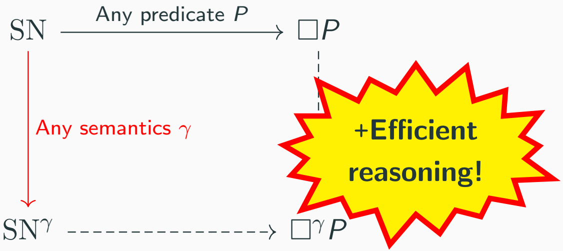
(1) and (3) are trivial, (2) is straightforward once you realize that $\Box Q$ is an **invariant** w.r.t. $\rightarrow$ for all $Q$. $\Box$

# Objective Complete

## Let’s explore the other direction



Let's explore the other direction



# The (vanilla) abstract setting of Logical Predicates

1. An operational semantics of a higher-order language is given.

Concrete/Abstract

2. A (type-indexed) predicate $P \rightarrowtail \mu\Sigma$ is given.

3. We construct a suitable *logical predicate over* $P$, *say* $\Box P$, *which implies* $P$.

# The (vanilla) abstract setting of Logical Predicates

1. An operational semantics of a higher-order language is given.

Concrete/Abstract

- (The model generated by) Operational Rules $\frac{t \to t'}{t \cdot s \to t' \cdot s}$

2. A (type-indexed) predicate $P \rightarrowtail \mu\Sigma$ is given.

3. We construct a suitable *logical predicate over $P$, say $\Box P$, which implies $P$.*

# The (vanilla) abstract setting of Logical Predicates

1. An operational semantics of a higher-order language is given.

**Concrete/Abstract**

- (The model generated by) Operational Rules $\frac{t \to t'}{t \cdot s \to t' \cdot s}$

i. Coalgebra $\gamma \colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$,

2. A (type-indexed) predicate $P \rightarrowtail \mu\Sigma$ is given.

3. We construct a suitable *logical predicate over* $P$*, say* $\Box P$*, which implies* $P$.

# The (vanilla) abstract setting of Logical Predicates

1. An operational semantics of a higher-order language is given.

**Concrete/Abstract**

- (The model generated by) Operational Rules $\frac{t \to t'}{t \cdot s \to t' \cdot s}$

i. Coalgebra $\gamma \colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$,

ii. on initial algebra $\iota \colon \Sigma\mu\Sigma \to \mu\Sigma$.

2. A (type-indexed) predicate $P \rightarrowtail \mu\Sigma$ is given.

3. We construct a suitable *logical predicate over* $P$*, say* $\Box P$*, which implies* $P$.

# The (vanilla) abstract setting of Logical Predicates

1. An operational semantics of a higher-order language is given.

**Concrete/Abstract**

| Concrete | Abstract |
|---|---|
| - (The model generated by) Operational Rules $\frac{t \to t'}{t \cdot s \to t' \cdot s}$ | i. Coalgebra $\gamma \colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$,<br>ii. on initial algebra $\iota \colon \Sigma\mu\Sigma \to \mu\Sigma$. |

2. A (type-indexed) predicate $P \rightarrowtail \mu\Sigma$ is given.

| Concrete | Abstract |
|---|---|
| - Family $(P_\tau \subseteq \Lambda_\tau)_{\tau \in \mathsf{Ty}}$ | |

3. We construct a suitable *logical predicate over* $P$, *say* $\Box P$, *which implies* $P$.

# The (vanilla) abstract setting of Logical Predicates

1. An operational semantics of a higher-order language is given.

**Concrete/Abstract**

| Concrete | Abstract |
| --- | --- |
| - (The model generated by) Operational Rules $\frac{t \to t'}{t \cdot s \to t' \cdot s}$ | i. Coalgebra $\gamma \colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$,<br>ii. on initial algebra $\iota \colon \Sigma\mu\Sigma \to \mu\Sigma$. |

2. A (type-indexed) predicate $P \rightarrowtail \mu\Sigma$ is given.

| Concrete | Abstract |
| --- | --- |
| - Family $(P_\tau \subseteq \Lambda_\tau)_{\tau \in \mathsf{Ty}}$ | - Monomorphism $P \rightarrowtail \mu\Sigma$ |

3. We construct a suitable *logical predicate over $P$, say $\square P$, which implies $P$*.

# The (vanilla) abstract setting of Logical Predicates

1. An operational semantics of a higher-order language is given.

**Concrete/Abstract**

| Concrete | Abstract |
|---|---|
| - (The model generated by) Operational Rules $\frac{t \to t'}{t \cdot s \to t' \cdot s}$ | i. Coalgebra $\gamma \colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$,<br>ii. on initial algebra $\iota \colon \Sigma\mu\Sigma \to \mu\Sigma$. |

2. A (type-indexed) predicate $P \rightarrowtail \mu\Sigma$ is given.

| Concrete | Abstract |
|---|---|
| - Family $(P_\tau \subseteq \Lambda_\tau)_{\tau \in \mathsf{Ty}}$ | - Monomorphism $P \rightarrowtail \mu\Sigma$ |

3. We construct a suitable *logical predicate over* $P$*, say* $\Box P$*, which implies* $P$.

| Concrete | Abstract |
|---|---|
| - Empirical, mysterious, problem-specific logical predicate $\mathrm{SN}$ | |

# The (vanilla) abstract setting of Logical Predicates

1. An operational semantics of a higher-order language is given.

**Concrete/Abstract**

| Concrete | Abstract |
|---|---|
| - (The model generated by) Operational Rules $\frac{t \to t'}{t \cdot s \to t' \cdot s}$ | i. Coalgebra $\gamma \colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$,<br>ii. on initial algebra $\iota \colon \Sigma\mu\Sigma \to \mu\Sigma$. |

2. A (type-indexed) predicate $P \rightarrowtail \mu\Sigma$ is given.

| Concrete | Abstract |
|---|---|
| - Family $(P_\tau \subseteq \Lambda_\tau)_{\tau \in \mathsf{Ty}}$ | - Monomorphism $P \rightarrowtail \mu\Sigma$ |

3. We construct a suitable *logical predicate over $P$, say $\square P$, which implies $P$*.

| Concrete | Abstract |
|---|---|
| - Empirical, mysterious, problem-specific logical predicate $\mathrm{SN}$ | - Generic predicate transformer $\square^{\gamma,\overline{B}} \colon \mathrm{Pred}_{\mu\Sigma}(\mathcal{C}) \to \mathrm{Pred}_{\mu\Sigma}(\mathcal{C})$ |

# (Vanilla) Logical Predicates proof method in the abstract

Assuming the following:

1. An initial algebra (object of terms) $\Sigma\mu\Sigma \xrightarrow{\iota} \mu\Sigma$,
2. an "operational semantics" morphism $\mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$ for some bifunctor $B \colon \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathcal{C}$,
3. and logical predicates $\Box(-)$,

the proof method of logical predicates amount to the following:

**Fundamental Property**

As initial algebras have no proper subalgebras, then

$$\overline{\Sigma}(\Box P) \leq \iota^\star[\Box P] \implies \Box P \cong \mu\Sigma \implies P \cong \mu\Sigma.$$

# Categorical machinery

$$B(X,Y) : \mathcal{C}^{\mathsf{op}} \times \mathcal{C} \to \mathcal{C} \quad \gamma \colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$$

$$B(X,Y) = Y + Y^X \qquad \gamma(t) = t' \text{ if } t \to t' \text{ and } \gamma(\lambda x.M) = (e \mapsto M[e/x])$$

# Categorical machinery

$$B(X,Y) : \mathcal{C}^{\mathsf{op}} \times \mathcal{C} \to \mathcal{C} \qquad \gamma \colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$$

$$B(X,Y) = Y + Y^X \qquad \gamma(t) = t' \text{ if } t \to t' \text{ and } \gamma(\lambda x.M) = (e \mapsto M[e/x])$$

$$\begin{array}{ccc} \mathrm{Pred}(\mathcal{C})^{\mathsf{op}} \times \mathrm{Pred}(\mathcal{C}) & \xrightarrow{\overline{B}} & \mathrm{Pred}(\mathcal{C}) \\ {\scriptstyle |-|^{\mathsf{op}} \times |-|}\Big\downarrow & & \Big\downarrow{\scriptstyle |-|} \\ \mathcal{C}^{\mathsf{op}} \times \mathcal{C} & \xrightarrow{\quad B \quad} & \mathcal{C} \end{array}$$

# Categorical machinery

$$B(X,Y) : \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathcal{C} \qquad \gamma\colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$$

$$B(X,Y) = Y + Y^X \qquad \gamma(t) = t' \text{ if } t \to t' \text{ and } \gamma(\lambda x.M) = (e \mapsto M[e/x])$$

$$\begin{array}{ccc} \mathrm{Pred}(\mathcal{C})^{\mathrm{op}} \times \mathrm{Pred}(\mathcal{C}) & \xrightarrow{\overline{B}} & \mathrm{Pred}(\mathcal{C}) \\ {\scriptstyle |-|^{\mathrm{op}} \times |-|}\downarrow & & \downarrow{\scriptstyle |-|} \\ \mathcal{C}^{\mathrm{op}} \times \mathcal{C} & \xrightarrow{B} & \mathcal{C} \end{array}$$

For example, $\overline{B}(P,Q) \subseteq \mu\Sigma + \mu\Sigma^{\mu\Sigma}$ is the disjoint union of (i) the set $\{t \mid Q(t)\}$ and (ii) the set of functions $f \in \mu\Sigma^{\mu\Sigma}$ that map inputs in $P$ to outputs in $Q$.

# Logical Predicates

**Relative invariant**

Let $c\colon Y \to B(X, Y)$ be a $B(X, -)$-coalgebra. Given predicates $S \rightarrowtail X$, $P \rightarrowtail Y$, we say that $P$ is an $S$-relative ($\overline{B}$-)invariant (for $c$) if

$$P \le c^{\star}[\overline{B}(S, P)].$$

**Logical Predicate**

A predicate $P \rightarrowtail \mu\Sigma$ is logical (for $\gamma$) if it is a $P$-relative $\overline{B}$-invariant.

# Logical Predicates

**Relative invariant**

Let $c\colon Y \to B(X,Y)$ be a $B(X,-)$-coalgebra. Given predicates $S \rightarrowtail X$, $P \rightarrowtail Y$, we say that $P$ is an $S$-relative ($\overline{B}$-)invariant (for $c$) if

$$P \leq c^{\star}[\overline{B}(S,P)].$$

**Logical Predicate**

A predicate $P \rightarrowtail \mu\Sigma$ is logical (for $\gamma$) if it is a $P$-relative $\overline{B}$-invariant.

A predicate $P$ is logical if for all $t \in \mu\Sigma$, $P(t)$ implies:

# Logical Predicates

**Relative invariant**

Let $c\colon Y \to B(X, Y)$ be a $B(X, -)$-coalgebra. Given predicates $S \rightarrowtail X$, $P \rightarrowtail Y$, we say that $P$ is an $S$-relative ($\overline{B}$-)invariant (for $c$) if

$$P \leq c^{\star}[\overline{B}(S, P)].$$

**Logical Predicate**

A predicate $P \rightarrowtail \mu\Sigma$ is logical (for $\gamma$) if it is a $P$-relative $\overline{B}$-invariant.

A predicate $P$ is logical if for all $t \in \mu\Sigma$, $P(t)$ implies:

1. If $t \to t'$, then $P(t')$ (with ND: if $\exists t.\, t \to t'$, then $P(t')$).

# Logical Predicates

## Relative invariant

Let $c \colon Y \to B(X, Y)$ be a $B(X, -)$-coalgebra. Given predicates $S \rightarrowtail X$, $P \rightarrowtail Y$, we say that $P$ is an $S$-relative ($\overline{B}$-)invariant (for $c$) if

$$P \leq c^\star[\overline{B}(S, P)].$$

## Logical Predicate

A predicate $P \rightarrowtail \mu\Sigma$ is logical (for $\gamma$) if it is a $P$-relative $\overline{B}$-invariant.

A predicate $P$ is logical if for all $t \in \mu\Sigma$, $P(t)$ implies:

1. If $t \to t'$, then $P(t')$ (with ND: if $\exists t.\, t \to t'$, then $P(t')$).
2. For all $s$, if $t \xrightarrow{s} t'$ and $P(s)$, then $P(t')$.

# One logical predicate to rule them all

## The $\Box$

Under certain conditions, the most important being that the predicate lifting $\overline{B}$ is **predicate-contractive**, for every predicate $P \rightarrowtail X$ on the state space of our coalgebra $X \to B(X, X)$ (i.e. a program property), there exists a certain "large" predicate $\Box P$ such that:

1. $\Box P \leq P$
2. $\Box P \leq c^\star[\overline{B}(\Box P, \Box P)]$ (i.e. $\Box P$ is logical)
3. $\Box P$ is the largest $\Box P$-relative invariant.

# One logical predicate to rule them all

**The $\Box$**

Under certain conditions, the most important being that the predicate lifting $\overline{B}$ is **predicate-contractive**, for every predicate $P \rightarrowtail X$ on the state space of our coalgebra $X \to B(X, X)$ (i.e. a program property), there exists a certain "large" predicate $\Box P$ such that:

1. $\Box P \leq P$
2. $\Box P \leq c^{\star}[\overline{B}(\Box P, \Box P)]$ (i.e. $\Box P$ is logical)
3. $\Box P$ is the largest $\Box P$-relative invariant.

**Conclusion/translation:** The lifting being defined inductively on types is sufficient for the existence of this magical, suitable logical predicate.

# Induction up to □

The definition of logicality and □ systematizes the logical predicates proof method, but where is the "efficient reasoning"?

# Induction up to $\Box$

The definition of logicality and $\Box$ systematizes the logical predicates proof method, but where is the "efficient reasoning"?

**Induction up to $\Box$**

For a certain class of **higher-order GSOS laws**, instead of laboriously showing $\overline{\Sigma}(\Box P) \leq \iota^{\star}[\Box P]$, it suffices to show the much simpler $\overline{\Sigma}(\Box P) \leq \iota^{\star}[P]$.

# Induction up to $\Box$

The definition of logicality and $\Box$ systematizes the logical predicates proof method, but where is the "efficient reasoning"?

**Induction up to $\Box$**

For a certain class of **higher-order GSOS laws**, instead of laboriously showing $\overline{\Sigma}(\Box P) \leq \iota^\star[\Box P]$, it suffices to show the much simpler $\overline{\Sigma}(\Box P) \leq \iota^\star[P]$.

Note: *Things are a bit more complex in languages with binding and substitution due to contractivity considerations, but the principle is the same.*

# Induction up to $\Box$

The definition of logicality and $\Box$ systematizes the logical predicates proof method, but where is the "efficient reasoning"?

**Induction up to $\Box$**

For a certain class of **higher-order GSOS laws**, instead of laboriously showing $\overline{\Sigma}(\Box P) \leq \iota^\star[\Box P]$, it suffices to show the much simpler $\overline{\Sigma}(\Box P) \leq \iota^\star[P]$.

Note: *Things are a bit more complex in languages with binding and substitution due to contractivity considerations, but the principle is the same. This explains the need to extend the predicate to open terms.*

**Induction up to $\boxdot$**

For a certain class of $\lambda$-**laws**, instead of laboriously showing $\overline{\Sigma}(\boxdot P) \leq \iota^\star[\boxdot P]$, it suffices to show the much simpler $\overline{\Sigma}(\Box P) \leq \iota^\star[P]$.

Thank you!