

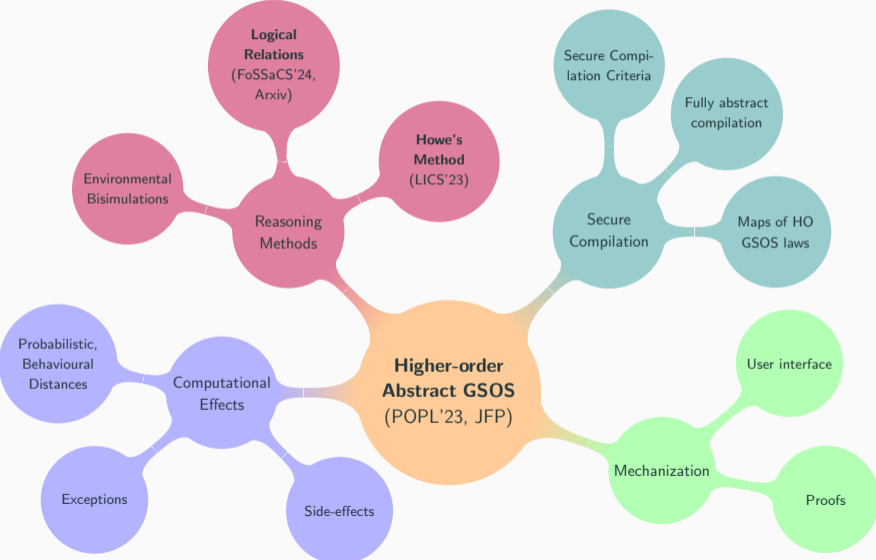
Logical Predicates

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 λ -calculus.
 - Write $\Lambda_\tau(\Gamma)$ for the set $\{t \mid \Gamma \vdash t : \tau\}$ and Λ_τ for the set $\{t \mid \emptyset \vdash t : \tau\}$.

The setting of Logical Predicates

1. An operational semantics of a higher-order language
 - Typically a typed λ -calculus.
 - Write $\Lambda_\tau(\Gamma)$ for the set $\{t \mid \Gamma \vdash t : \tau\}$ and Λ_τ for the set $\{t \mid \emptyset \vdash t : \tau\}$.
2. A (type-indexed) predicate $P \rightsquigarrow \Lambda$, that can't be proven inductively
 - Family $(P_\tau \subseteq \Lambda_\tau)_{\tau \in \mathbb{T}_y}$
 - Strong normalization, type safety etc.

The setting of Logical Predicates

1. An operational semantics of a higher-order language
 - Typically a typed λ -calculus.
 - Write $\Lambda_\tau(\Gamma)$ for the set $\{t \mid \Gamma \vdash t : \tau\}$ and Λ_τ for the set $\{t \mid \emptyset \vdash t : \tau\}$.
2. A (type-indexed) predicate $P \rightsquigarrow \Lambda$, 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 λ -calculus.
 - Write $\Lambda_\tau(\Gamma)$ for the set $\{t \mid \Gamma \vdash t : \tau\}$ and Λ_τ for the set $\{t \mid \emptyset \vdash t : \tau\}$.
2. A (type-indexed) predicate $P \rightsquigarrow \Lambda$, 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.

Definition (A standard logical predicate)

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

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

Definition (A standard logical predicate)

$$\begin{aligned} \text{SN}_{\text{unit}}(t) &= \Downarrow_{\text{unit}}(t) \\ \text{SN}_{\tau_1 \rightarrow \tau_2}(t) &= \Downarrow_{\tau_1 \rightarrow \tau_2}(t) \wedge (\forall s: \tau_1. \text{SN}_{\tau_1}(s) \implies \text{SN}_{\tau_2}(t \cdot s)) \end{aligned}$$

Definition (Open extension of SN)

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

Strong Normalization

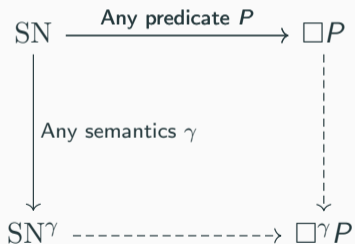
One annoying case of the proof is that of λ -abstraction $\Gamma \vdash \lambda x: \tau_1. t: \tau_1 \rightarrow \tau_2$.

Given a substitution $(\emptyset \vdash e_n: \Gamma(n))_{n \in |\Gamma|}$ satisfying SN, we have to:

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

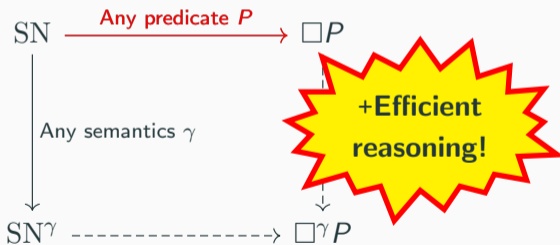
Complex language \implies complex argument...

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)

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

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

Dissecting the logical predicate (1)

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

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

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

Dissecting the logical predicate (1)

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

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

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

$$\Downarrow\Downarrow_{\text{unit}}(t) = \Downarrow_{\text{unit}}(t)$$

$$\Downarrow\Downarrow_{\tau_1 \rightarrow \tau_2}(t) = \Downarrow_{\tau_1 \rightarrow \tau_2} t \wedge (\forall s: \tau_1. t \xRightarrow{s} t' \wedge \Downarrow\Downarrow_{\tau_1}(s) \implies \Downarrow\Downarrow_{\tau_2}(t'))$$

Dissecting the logical predicate (1)

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

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

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

$$\Downarrow\!\!\Downarrow_{\text{unit}}(t) = \Downarrow_{\text{unit}}(t)$$

$$\Downarrow\!\!\Downarrow_{\tau_1 \rightarrow \tau_2}(t) = \Downarrow_{\tau_1 \rightarrow \tau_2} t \wedge (\forall s: \tau_1. t \xrightarrow{s} t' \wedge \Downarrow\!\!\Downarrow_{\tau_1}(s) \implies \Downarrow\!\!\Downarrow_{\tau_2}(t'))$$

Idea : Abstract away from the predicate \Downarrow

Dissecting the logical predicate (2)

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

$$\Box P_{\tau_1 \rightarrow \tau_2}(t) = P_{\tau_1 \rightarrow \tau_2} t \wedge (\forall s: \tau_1. t \xrightarrow{s} t' \wedge \Box P_{\tau_1}(s) \implies \Box P_{\tau_2}(t'))$$

Dissecting the logical predicate (2)

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

$$\Box P_{\tau_1 \rightarrow \tau_2}(t) = P_{\tau_1 \rightarrow \tau_2} t \wedge (\forall s: \tau_1. t \xrightarrow{s} t' \wedge \Box P_{\tau_1}(s) \implies \Box P_{\tau_2}(t'))$$

Idea : Move one from \implies to the more fundamental \rightarrow

Dissecting the logical predicate (2)

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

$$\Box P_{\tau_1 \rightarrow \tau_2}(t) = P_{\tau_1 \rightarrow \tau_2} t \wedge (\forall s: \tau_1. t \xrightarrow{s} t' \wedge \Box P_{\tau_1}(s) \implies \Box P_{\tau_2}(t'))$$

Idea : Move one from \implies to the more fundamental \rightarrow

greatest subset of $\wedge_{\tau_1 \rightarrow \tau_2}$

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

$$\Box P_{\tau_1 \rightarrow \tau_2}(t) \implies P_{\tau_1 \rightarrow \tau_2}(t) \wedge \begin{cases} \Box P_{\tau_1 \rightarrow \tau_2}(t') & \text{if } t \rightarrow t' \\ \Box P_{\tau_1}(s) \implies \Box P_{\tau_2}(t') & \text{if } t \xrightarrow{s} t' \end{cases}$$

Induction up to \Box on STLC

Theorem

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

1. the unit expression $e: \text{unit}$ satisfies $\Box_{\text{unit}} P$ P_{unit} ,
2. for all closed application terms $t s$ such that $\Box_{\tau_1 \rightarrow \tau_2} P(t)$ and $\Box_{\tau_1} P(s)$, we have $\Box_{\tau_2} P(t s)$ $P_{\tau_2}(t s)$, and
3. for all λ -abstractions $\lambda x: \tau_1. t: \tau_1 \rightarrow \tau_2$, 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 $(\text{Th36}.P)_\tau(\emptyset) = P_\tau$ and $(\text{Th36}.P)_\tau(\Gamma \neq \emptyset) = \top$. \square

Let's try this out!

Proving strong normalization for STLC

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

Let's try this out!

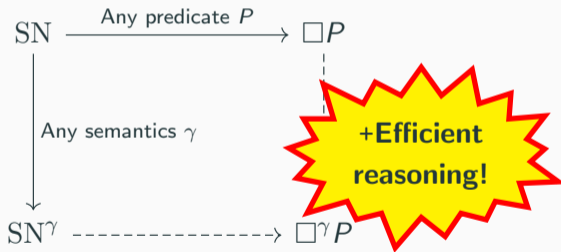
Proving strong normalization for STLC

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

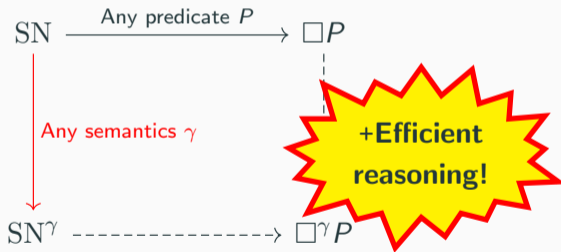
Proof.

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

Let's explore the other direction



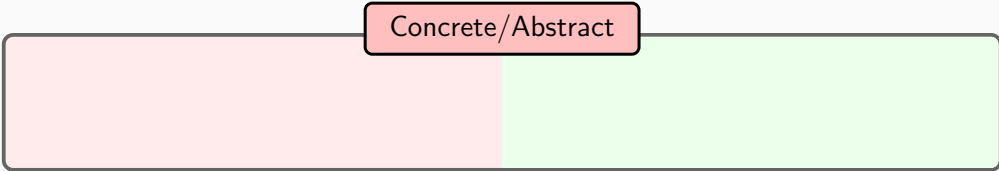
Let's explore the other direction



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 $P \rightsquigarrow \mu\Sigma$ 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 $\frac{t \rightarrow t'}{t \cdot s \rightarrow t' \cdot s}$

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

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

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

i. Coalgebra $\gamma: \mu\Sigma \rightarrow B(\mu\Sigma, \mu\Sigma)$,

2. A (type-indexed) predicate $P \rightsquigarrow \mu\Sigma$ 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 $\frac{t \rightarrow t'}{t \cdot s \rightarrow t' \cdot s}$

- i. Coalgebra $\gamma: \mu\Sigma \rightarrow B(\mu\Sigma, \mu\Sigma)$,
- ii. on initial algebra $\iota: \Sigma\mu\Sigma \rightarrow \mu\Sigma$.

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

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

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

- i. Coalgebra $\gamma: \mu\Sigma \rightarrow B(\mu\Sigma, \mu\Sigma)$,
- ii. on initial algebra $\iota: \Sigma\mu\Sigma \rightarrow \mu\Sigma$.

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

- Family $(P_\tau \subseteq \Lambda_\tau)_{\tau \in \text{Ty}}$

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

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

- i. Coalgebra $\gamma: \mu\Sigma \rightarrow B(\mu\Sigma, \mu\Sigma)$,
- ii. on initial algebra $\iota: \Sigma\mu\Sigma \rightarrow \mu\Sigma$.

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

- Family $(P_\tau \subseteq \Lambda_\tau)_{\tau \in \text{Ty}}$

- Monomorphism $P \rightsquigarrow \mu\Sigma$

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

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

- i. Coalgebra $\gamma: \mu\Sigma \rightarrow B(\mu\Sigma, \mu\Sigma)$,
- ii. on initial algebra $\iota: \Sigma\mu\Sigma \rightarrow \mu\Sigma$.

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

- Family $(P_\tau \subseteq \Lambda_\tau)_{\tau \in \text{Ty}}$

- Monomorphism $P \rightsquigarrow \mu\Sigma$

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

- Empirical, mysterious, problem-specific logical predicate SN

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 $\frac{t \rightarrow t'}{t \cdot s \rightarrow t' \cdot s}$

- i. Coalgebra $\gamma: \mu\Sigma \rightarrow B(\mu\Sigma, \mu\Sigma)$,
- ii. on initial algebra $\iota: \Sigma\mu\Sigma \rightarrow \mu\Sigma$.

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

- Family $(P_\tau \subseteq \Lambda_\tau)_{\tau \in \text{Ty}}$

- Monomorphism $P \rightsquigarrow \mu\Sigma$

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

- Empirical, mysterious, problem-specific logical predicate SN

- Generic predicate transformer
 $\square\gamma, \bar{B}: \text{Pred}_{\mu\Sigma}(\mathcal{C}) \rightarrow \text{Pred}_{\mu\Sigma}(\mathcal{C})$

(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 $\mu\Sigma \rightarrow B(\mu\Sigma, \mu\Sigma)$ for some bifunctor $B: \mathcal{C}^{\text{op}} \times \mathcal{C} \rightarrow \mathcal{C}$,
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

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

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

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

Categorical machinery

$$B(X, Y) : \mathcal{C}^{\text{op}} \times \mathcal{C} \rightarrow \mathcal{C} \quad \gamma : \mu\Sigma \rightarrow B(\mu\Sigma, \mu\Sigma)$$
$$B(X, Y) = Y + Y^X \quad \gamma(t) = t' \text{ if } t \rightarrow t' \text{ and } \gamma(\lambda x.M) = (e \mapsto M[e/x])$$

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

Categorical machinery

$$B(X, Y) : \mathcal{C}^{\text{op}} \times \mathcal{C} \rightarrow \mathcal{C} \quad \gamma : \mu\Sigma \rightarrow B(\mu\Sigma, \mu\Sigma)$$
$$B(X, Y) = Y + Y^X \quad \gamma(t) = t' \text{ if } t \rightarrow t' \text{ and } \gamma(\lambda x.M) = (e \mapsto M[e/x])$$

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

For example, $\bar{B}(P, Q) \subseteq \mu\Sigma + \mu\Sigma^{\mu\Sigma}$ 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 .

Relative invariant

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

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

Logical Predicate

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

Relative invariant

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

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

Logical Predicate

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

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

Relative invariant

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

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

Logical Predicate

A predicate $P \rightsquigarrow \mu\Sigma$ is logical (for γ) 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 $t \rightarrow t'$, then $P(t')$ (with ND: if $\exists t. t \rightarrow t'$, then $P(t')$).

Relative invariant

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

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

Logical Predicate

A predicate $P \rightsquigarrow \mu\Sigma$ is logical (for γ) 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 $t \rightarrow t'$, then $P(t')$ (with ND: if $\exists t. t \rightarrow t'$, 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 \bar{B} is **predicate-contractive**, for every predicate $P \multimap X$ on the state space of our coalgebra $X \rightarrow B(X, X)$ (i.e. a program property), there exists a certain “large” predicate $\Box P$ such that:

1. $\Box P \leq P$
2. $\Box P \leq c^*[\bar{B}(\Box P, \Box P)]$ (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 $P \rightsquigarrow X$ on the state space of our coalgebra $X \rightarrow B(X, X)$ (i.e. a program property), there exists a certain “large” predicate $\Box P$ such that:

1. $\Box P \leq P$
2. $\Box P \leq c^*[\overline{B}(\Box P, \Box P)]$ (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 \square

The definition of logicality and \square 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 $\bar{\Sigma}(\Box P) \leq \iota^*[\Box P]$, it suffices to show the much simpler $\bar{\Sigma}(\Box P) \leq \iota^*[P]$.

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 $\bar{\Sigma}(\Box P) \leq \iota^*[\Box P]$, it suffices to show the much simpler $\bar{\Sigma}(\Box P) \leq \iota^*[P]$.

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 $\bar{\Sigma}(\Box P) \leq \iota^*[\Box P]$, it suffices to show the much simpler $\bar{\Sigma}(\Box P) \leq \iota^*[P]$.

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 \Box

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

Thank you!