Logical Relations (and more)

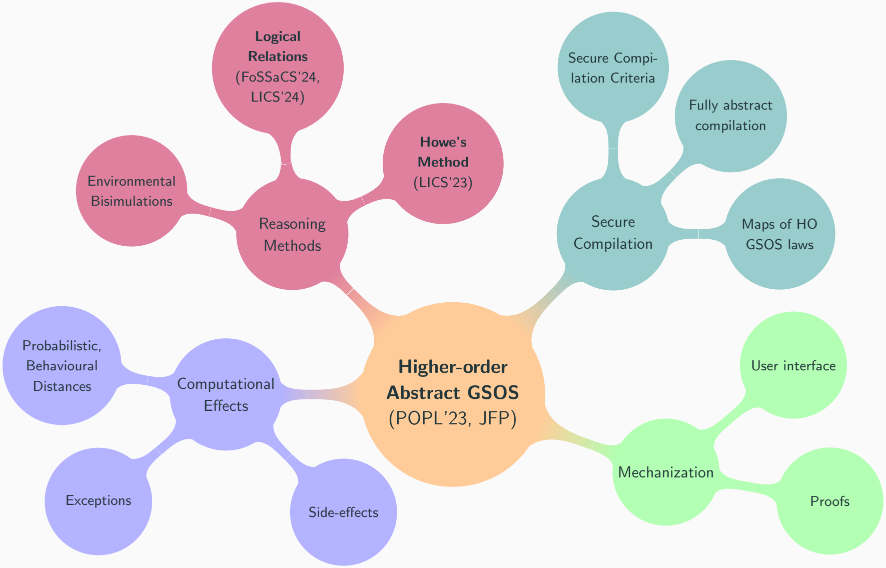
in Higher-order Mathematical Operational Semantics

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

Chocola, May 2024

Friedrich-Alexander-Universität Erlangen-Nürnberg

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



Overview

HO-MOS or
Higher-order Abstract GSOS

Relational Reasoning

Step-indexed Logical Relations

Logical Predicates

HO-MOS or Higher-order Abstract GSOS

GSOS rules

Definition (GSOS rule)

$$\frac{\{x_i \xrightarrow{a} y^a_{ij}\}^{1 \leq i \leq \mathsf{ar}(\mathsf{f}),\, a \in A_i}_{1 \leq j \leq n^a_i} \qquad \{x_i \stackrel{b}{\not\rightarrow}\}^{1 \leq i \leq \mathsf{ar}(\mathsf{f})}_{b \in B_i}}{\mathsf{f}(x_1, \ldots, x_{\mathsf{ar}(\mathsf{f})}) \xrightarrow{c} t}$$

where $\mathsf{f} \in \overline{\Sigma}$, A_i, B_i range over subsets of L and $n^a_i \in \mathbb{N}$ and $c \in \mathsf{L}$. Variables x_i and y^a_{ij} are all distinct and are the only variables appearing in t.

GSOS rules

Definition (GSOS rule)

$$\frac{\{x_i \xrightarrow{a} y_{ij}^a\}_{1 \le j \le n_i^a}^{1 \le i \le \mathrm{ar}(\mathsf{f}),\, a \in A_i} \qquad \{x_i \stackrel{b}{\not\to}\}_{b \in B_i}^{1 \le i \le \mathrm{ar}(\mathsf{f})}}{\mathsf{f}(x_1, \ldots, x_{\mathrm{ar}(\mathsf{f})}) \xrightarrow{c} t}$$

where $\mathsf{f} \in \overline{\Sigma}$, A_i, B_i range over subsets of L and $n_i^a \in \mathbb{N}$ and $c \in$ L. Variables x_i and y_{ij}^a are all distinct and are the only variables appearing in t.

Example rule

$$\frac{p \xrightarrow{a} p'}{p \,||\, q \xrightarrow{a} p' \,||\, q}$$

GSOS rules

Definition (GSOS rule)

$$\frac{\{x_i \xrightarrow{a} y_{ij}^a\}_{1 \le j \le n_i^a}^{1 \le i \le \mathrm{ar}(\mathsf{f}),\, a \in A_i} \qquad \{x_i \stackrel{b}{\not\to}\}_{b \in B_i}^{1 \le i \le \mathrm{ar}(\mathsf{f})}}{\mathsf{f}(x_1, \ldots, x_{\mathrm{ar}(\mathsf{f})}) \xrightarrow{c} t}$$

where $\mathsf{f} \in \overline{\Sigma}$, A_i, B_i range over subsets of L and $n_i^a \in \mathbb{N}$ and $c \in$ L. Variables x_i and y_{ij}^a are all distinct and are the only variables appearing in t.

Example rule

generic generic

$$\frac{p \xrightarrow{a} p'}{p \,||\, q \xrightarrow{a} p' \,||\, q}$$

Abstract GSOS (Mathematical Operational Semantics)

Let endofunctors $\Sigma, B\colon \mathcal{C} \to \mathcal{C}$ in some distributive category $\mathcal{C}$ and assume that the free monad over Σ, $\Sigma^\star$, exists.

Definition (Turi and Plotkin [1])

A GSOS law of Σ (modelling the syntax of the system) over B (modelling the behaviour) is a natural transformation [1]

$$\rho_X\colon \Sigma(X \times BX) \to B\Sigma^\star X.$$

[1] Roughly a parametrically polymorphic function.

Abstract GSOS

Theorem (Turi and Plotkin [1])

Let finitary signature $\overline{\Sigma}$, with associated endofunctor $\Sigma \colon \mathsf{Set} \to \mathsf{Set}$, and a finite set of actions L. GSOS specifications of $\overline{\Sigma}$ over L are in a bijective correspondence with GSOS laws of Σ over $(\mathcal{P}_\mathsf{f} X)^\mathsf{L}$.

$$\frac{p \xrightarrow{a} p'}{p \,||\, q \xrightarrow{a} p' \,||\, q}$$

$$\cong$$

$$\rho_X \colon \coprod_{\mathsf{f} \in \overline{\Sigma}} (X \times (\mathcal{P}_\mathsf{f} X)^\mathsf{L})^{\mathsf{ar}(\mathsf{f})} \to (\mathcal{P}_\mathsf{f} \Sigma^* X)^\mathsf{L}$$

Abstract GSOS

Theorem (Turi and Plotkin [1])

Let finitary signature $\overline{\Sigma}$, with associated endofunctor $\Sigma\colon \mathsf{Set} \to \mathsf{Set}$, and a finite set of actions L. GSOS specifications of $\overline{\Sigma}$ over L are in a bijective correspondence with GSOS laws of Σ over $(\mathcal{P}_{\mathsf{f}} X)^{\mathsf{L}}$.

$$\frac{p \xrightarrow{a} p'}{p \,||\, q \xrightarrow{a} p' \,||\, q}$$

$$\cong$$

$$\rho_X \colon \coprod_{\mathsf{f} \in \overline{\Sigma}} (X \times (\mathcal{P}_{\mathsf{f}} X)^{\mathsf{L}})^{\mathsf{ar(f)}} \to (\mathcal{P}_{\mathsf{f}} \Sigma^* X)^{\mathsf{L}}$$

Abstract GSOS

Theorem (Turi and Plotkin [1])

Let finitary signature $\overline{\Sigma}$, with associated endofunctor $\Sigma\colon \mathsf{Set} \to \mathsf{Set}$, and a finite set of actions L. GSOS specifications of $\overline{\Sigma}$ over L are in a bijective correspondence with GSOS laws of Σ over $(\mathcal{P}_{\mathsf{f}} X)^{\mathsf{L}}$.

$$\frac{p \xrightarrow{a} p'}{p \,||\, q \xrightarrow{a} p' \,||\, q}$$

$$\cong$$

$$\rho_X \colon \coprod_{\mathsf{f} \in \overline{\Sigma}} (X \times (\mathcal{P}_{\mathsf{f}} X)^{\mathsf{L}})^{\mathsf{ar}(\mathsf{f})} \to (\mathcal{P}_{\mathsf{f}} \Sigma^* X)^{\mathsf{L}}$$

Abstract GSOS

Theorem (Turi and Plotkin [1])

Let finitary signature $\overline{\Sigma}$, with associated endofunctor $\Sigma\colon \mathsf{Set} \to \mathsf{Set}$, and a finite set of actions L. GSOS specifications of $\overline{\Sigma}$ over L are in a bijective correspondence with GSOS laws of Σ over $(\mathcal{P}_{\mathsf{f}} X)^{\mathsf{L}}$.

$$\frac{p \xrightarrow{a} p'}{p \mid\mid q \xrightarrow{a} p' \mid\mid q}$$

$$\cong$$

$$\rho_X\colon \coprod_{\mathsf{f}\in\overline{\Sigma}} (\, X \; \times (\mathcal{P}_{\mathsf{f}} \; X \;)^{\mathsf{L}})^{\mathsf{ar}(\mathsf{f})} \to (\mathcal{P}_{\mathsf{f}} \Sigma^* \; X \;)^{\mathsf{L}}$$

Abstract GSOS

Theorem (Turi and Plotkin [1])

Let finitary signature $\overline{\Sigma}$, with associated endofunctor $\Sigma\colon \mathsf{Set} \to \mathsf{Set}$, and a finite set of actions L. GSOS specifications of $\overline{\Sigma}$ over L are in a bijective correspondence with GSOS laws of Σ over $(\mathcal{P}_\mathsf{f} X)^\mathsf{L}$.

$$\frac{\boxed{p} \xrightarrow{a} p'}{\boxed{p} \mid\mid \boxed{q} \xrightarrow{a} p' \mid\mid q}$$

$$\cong$$

$$\rho_X \colon \coprod_{\mathsf{f} \in \overline{\Sigma}} (\boxed{X} \times (\mathcal{P}_\mathsf{f}\ X\ )^\mathsf{L})^{\mathsf{ar}(\mathsf{f})} \to (\mathcal{P}_\mathsf{f} \Sigma^*\ X\ )^\mathsf{L}$$

Abstract GSOS

Theorem (Turi and Plotkin [1])

Let finitary signature $\overline{\Sigma}$, with associated endofunctor $\Sigma\colon \mathsf{Set} \to \mathsf{Set}$, and a finite set of actions L. GSOS specifications of $\overline{\Sigma}$ over L are in a bijective correspondence with GSOS laws of Σ over $(\mathcal{P}_{\mathsf{f}}X)^{\mathsf{L}}$.

$$\frac{\boxed{p} \xrightarrow{a} \boxed{p'}}{\boxed{p} \mid\mid \boxed{q} \xrightarrow{a} p' \mid\mid q}$$

$$\cong$$

$$\rho_X \colon \coprod_{\mathsf{f}\in\overline{\Sigma}} (\boxed{X} \times (\mathcal{P}_{\mathsf{f}} \boxed{X})^{\mathsf{L}})^{\mathsf{ar}(\mathsf{f})} \to (\mathcal{P}_{\mathsf{f}}\Sigma^{*} X)^{\mathsf{L}}$$

Abstract GSOS

Theorem (Turi and Plotkin [1])

Let finitary signature $\overline{\Sigma}$, with associated endofunctor $\Sigma\colon \mathsf{Set} \to \mathsf{Set}$, and a finite set of actions L. GSOS specifications of $\overline{\Sigma}$ over L are in a bijective correspondence with GSOS laws of Σ over $(\mathcal{P}_{\mathsf{f}} X)^{\mathsf{L}}$.

$$\frac{\boxed{p} \xrightarrow{a} \boxed{p'}}{\boxed{p} \,||\, \boxed{q} \xrightarrow{a} p' \,||\, q}$$

$$\cong$$

$$\rho_X \colon \coprod_{\mathsf{f} \in \overline{\Sigma}} (\boxed{X} \times (\mathcal{P}_{\mathsf{f}} \boxed{X})^{\mathsf{L}})^{\mathsf{ar}(\mathsf{f})} \to (\mathcal{P}_{\mathsf{f}} \Sigma^* \boxed{X})^{\mathsf{L}}$$

Abstract GSOS

Theorem (Turi and Plotkin [1])

Let finitary signature $\overline{\Sigma}$, with associated endofunctor $\Sigma\colon \mathsf{Set} \to \mathsf{Set}$, and a finite set of actions L. GSOS specifications of $\overline{\Sigma}$ over L are in a bijective correspondence with GSOS laws of Σ over $(\mathcal{P}_{\mathsf{f}} X)^{\mathsf{L}}$.

$$\frac{p \xrightarrow{a} p'}{p \mid\mid q \xrightarrow{a} p' \mid\mid q}$$

$$\cong$$

$$\rho_X \colon \coprod_{\mathsf{f} \in \overline{\Sigma}} (X \times (\mathcal{P}_{\mathsf{f}} X)^{\mathsf{L}})^{\mathsf{ar}(\mathsf{f})} \to (\mathcal{P}_{\mathsf{f}} \Sigma^* X)^{\mathsf{L}}$$

Abstract GSOS

The fascinating part is that GSOS laws gave a precise, concise mathematical representation of what GSOS specifications *are*.

Abstract GSOS

The fascinating part is that GSOS laws gave a precise, concise mathematical representation of what GSOS specifications *are*.

They are certain natural transformations.

Abstract GSOS

Operational rules

$$\frac{p \xrightarrow{a} p'}{p \,||\, q \xrightarrow{a} p' \,||\, q}$$

$\cong$

GSOS laws: natural transformations

$$\rho_X \colon \underbrace{\Sigma(X \times BX)}_{\text{premises}} \to \underbrace{B(\Sigma^* X)}_{\text{conclusion}}$$

for functors $\Sigma, B \colon \mathcal{C} \to \mathcal{C}$ representing syntax and behaviour (e.g. $B = \mathcal{P}_{\mathsf{f}}^{\mathsf{L}}$).

Abstract GSOS

Operational rules

$$\frac{p \xrightarrow{a} p'}{p \,||\, q \xrightarrow{a} p' \,||\, q}$$

$\cong$

GSOS laws: natural transformations

$$\rho_X\colon \underbrace{\Sigma(X \times BX)}_{\text{premises}} \to \underbrace{B(\Sigma^* X)}_{\text{conclusion}}$$

for functors $\Sigma, B\colon \mathcal{C} \to \mathcal{C}$ representing syntax and behaviour (e.g. $B = \mathcal{P}_{\mathsf{f}}^{\mathsf{L}}$).

(inductively defined) programs

(coinductive) behaviours

- Operational model $\mu\Sigma \to B(\mu\Sigma)$, denotational model $\Sigma(\nu B) \to \nu B$.

Abstract GSOS

Operational rules

$$\frac{p \xrightarrow{a} p'}{p \,||\, q \xrightarrow{a} p' \,||\, q}$$

$\cong$

GSOS laws: natural transformations

$$\rho_X \colon \underbrace{\Sigma(X \times BX)}_{\text{premises}} \to \underbrace{B(\Sigma^* X)}_{\text{conclusion}}$$

for functors $\Sigma, B \colon \mathcal{C} \to \mathcal{C}$ representing syntax and behaviour (e.g. $B = \mathcal{P}_\mathsf{f}^\mathsf{L}$).

(inductively defined) programs

(coinductive) behaviours

- Operational model $\mu\Sigma \to B(\mu\Sigma)$, denotational model $\Sigma(\nu B) \to \nu B$.
- **Key feature:** compositionality, i.e. bisimilarity is a congruence:

$$p_i \sim q_i \quad (i = 1, \ldots, n) \quad \xRightarrow{f \in \Sigma} \quad \mathsf{f}(p_1, \ldots, p_n) \sim \mathsf{f}(q_1, \ldots, q_n).$$

Abstract GSOS

Operational rules

$$\frac{p \xrightarrow{a} p'}{p \,||\, q \xrightarrow{a} p' \,||\, q}$$

$\cong$

GSOS laws: natural transformations

$$\rho_X \colon \underbrace{\Sigma(X \times BX)}_{\text{premises}} \to \underbrace{B(\Sigma^* X)}_{\text{conclusion}}$$

for functors $\Sigma, B \colon \mathcal{C} \to \mathcal{C}$ representing syntax and behaviour (e.g. $B = \mathcal{P}_{\mathrm{f}}^{\mathsf{L}}$).

(inductively defined) programs

(coinductive) behaviours

- Operational model $\mu\Sigma \to B(\mu\Sigma)$, denotational model $\Sigma(\nu B) \to \nu B$.
- **Key feature:** compositionality, i.e. bisimilarity is a congruence:

$$p_i \sim q_i \quad (i = 1, \ldots, n) \quad \xRightarrow{f \in \Sigma} \quad \mathrm{f}(p_1, \ldots, p_n) \sim \mathrm{f}(q_1, \ldots, q_n).$$

- **Scope**: first-order (CCS, π-calculus, ...), ~~higher-order~~ (λ-calculus, SKI calculus)

Higher-order abstract GSOS?

For all the success of abstract GSOS (variable binding [2], formats [3]–[6], effects [7]–[9], compilers [10]–[12]), higher-order languages have always been the big question mark.

An enduring problem

Turi and Plotkin 1997 [1]

The major challenge ahead is the operational semantics of the languages with variable binders, such as the π-calculus and the λ-calculus.

An enduring problem

Turi and Plotkin 1997 [1]

The major challenge ahead is the operational semantics of the languages with variable binders, such as the π-calculus and the λ-calculus.

[...]

An enduring problem

Turi and Plotkin 1997 [1]

The major challenge ahead is the operational semantics of the languages with variable binders, such as the π-calculus and the λ-calculus.

[...]

This approach has been deeply investigated, notably for quantitative languages [3]. However, as of today, attempts to apply it to higher-order (e.g., functional) languages have failed.

An enduring problem

Turi and Plotkin 1997 [1]

The major challenge ahead is the operational semantics of the languages with variable binders, such as the π-calculus and the λ-calculus.

[...]

Hirschowitz and Lafont 2022 [13]

This approach has been deeply investigated, notably for quantitative languages [3]. However, as of today, attempts to apply it to higher-order (e.g., functional) languages have failed.

$\mathrm{SKI_u}$ combinator calculus

$$\frac{}{S \xrightarrow{t} S'(t)} \qquad \frac{}{S'(p) \xrightarrow{t} S''(p, t)} \qquad \frac{}{S''(p, q) \xrightarrow{t} (p\, t)\, (q\, t)}$$

$$\frac{}{K \xrightarrow{t} K'(t)} \qquad \frac{}{K'(p) \xrightarrow{t} p} \qquad \frac{}{I \xrightarrow{t} t}$$

$$\frac{p \to p'}{p\, q \to p'\, q} \qquad \frac{p \xrightarrow{q} p'}{p\, q \to p'}$$

Figure 1: Small-step operational semantics of the $\mathrm{SKI_u}$ calculus, our version of the SKI combinator calculus, invented by Curry [14].

$\mathrm{SKI_u}$ combinator calculus

$$\frac{}{S \xrightarrow{t} S'(t)} \qquad \frac{}{S'(p) \xrightarrow{t} S''(p,t)} \qquad \frac{}{S''(p,q) \xrightarrow{t} (p\,t)\,(q\,t)}$$

$$\frac{}{K \xrightarrow{t} K'(t)} \qquad \frac{}{K'(p) \xrightarrow{t} p} \qquad \frac{}{I \xrightarrow{t} t}$$

$$\frac{p \to p'}{p\,q \to p'\,q} \qquad \frac{p \xrightarrow{q} p'}{p\,q \to p'}$$

Figure 1: Small-step operational semantics of the $\mathrm{SKI_u}$ calculus, our version of the SKI combinator calculus, invented by Curry [14].

Disclaimer: This is just a convenient example to introduce HO-MOS. The latter can do the λ-calculus, typed or untyped, with simple or recursive types, etc.

$\mathsf{SKI_u}$ **combinator calculus**

$$\frac{}{S''(p, q) \xrightarrow{t} (p\,t)\,(q\,t)} \qquad \frac{p \rightarrow p'}{p\,q \rightarrow p'\,q} \qquad \frac{p \xrightarrow{q} p'}{p\,q \rightarrow p'}$$

A combinator calculus

$$\frac{}{S''(p, q) \xrightarrow{t} (p\,t)\,(q\,t)} \qquad \frac{p \rightarrow p'}{p\,q \rightarrow p'\,q} \qquad \frac{p \xrightarrow{q} p'}{p\,q \rightarrow p'}$$

combinator

A combinator calculus

$$\frac{}{S''(p, q) \xrightarrow{t} (p\, t)\, (q\, t)} \qquad \frac{p \to p'}{p\, q \to p'\, q} \qquad \frac{p \xrightarrow{q} p'}{p\, q \to p'}$$

combinator

application

A combinator calculus

Labels can be a terms!

$$\frac{}{S''(p, q) \xrightarrow{t} (p\,t)\,(q\,t)} \qquad \frac{p \rightarrow p'}{p\,q \rightarrow p'\,q} \qquad \frac{p \xrightarrow{q} p'}{p\,q \rightarrow p'}$$

combinator

application

GSOS vs combinator calculi

GSOS

$$\frac{p \xrightarrow{a} p'}{p \,||\, q \xrightarrow{a} p' \,||\, q}$$

Is it GSOS?

$$\frac{p \rightarrow p'}{p\, q \rightarrow p'\, q}$$

$$\frac{p \xrightarrow{q} p'}{p\, q \rightarrow p'}$$

$$\frac{}{S''(p, q) \xrightarrow{t} (p\, t)\, (q\, t)}$$

GSOS vs combinator calculi

GSOS

$$\frac{p \xrightarrow{a} p'}{p \,||\, q \xrightarrow{a} p' \,||\, q}$$

Is it GSOS?

$$\frac{p \to p'}{p\, q \to p'\, q}$$ Yeah!

$$\frac{p \xrightarrow{q} p'}{p\, q \to p'}$$

$$\frac{}{S''(p, q) \xrightarrow{t} (p\, t)\,(q\, t)}$$

GSOS vs combinator calculi

GSOS

$$\frac{p \xrightarrow{a} p'}{p \,||\, q \xrightarrow{a} p' \,||\, q}$$

Is it GSOS?

$$\frac{p \rightarrow p'}{p\, q \rightarrow p'\, q}$$ Yeah!

$$\frac{p \xrightarrow{q} p'}{p\, q \rightarrow p'}$$ Nope!

$$\frac{}{S''(p, q) \xrightarrow{t} (p\, t)\,(q\, t)}$$

GSOS vs combinator calculi

GSOS

$$\frac{p \xrightarrow{a} p'}{p \,||\, q \xrightarrow{a} p' \,||\, q}$$

Is it GSOS?

$$\frac{p \rightarrow p'}{p\,q \rightarrow p'\,q}$$ Yeah!

$$\frac{p \xrightarrow{q} p'}{p\,q \rightarrow p'}$$ Nope!

$$\frac{}{S''(p, q) \xrightarrow{t} (p\,t)\,(q\,t)}$$ Nope!

The Issue With Higher-Order Languages

Higher-order languages require behaviours like

$$BX = X^X.$$

This is not an endofunctor – but

$$B(X, Y) = Y^X$$

is a bifunctor contravariant in X and covariant in Y.

The Issue With Higher-Order Languages

Higher-order languages require behaviours like

$$BX = X^X.$$

This is not an endofunctor – but

$$B(X, Y) = Y^X$$

is a bifunctor contravariant in X and covariant in Y.

Key idea for higher-order abstract GSOS

$$\text{endofunctors } B\colon \mathcal{C} \to \mathcal{C} \qquad + \qquad \text{natural transformations}$$

$$\Downarrow$$

$$\textbf{bifunctors } B\colon \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathcal{C} \qquad + \qquad \textbf{dinatural} \text{ transformations.}$$

Higher-order GSOS laws

Definition

A *higher-order GSOS law* of $\Sigma\colon \mathcal{C} \to \mathcal{C}$ (modelling the syntax) over $B\colon \mathcal{C}^{\mathsf{op}} \times \mathcal{C} \to \mathcal{C}$ (modelling higher-order behaviour) is a family of morphisms

$$\rho_{X,Y}\colon \Sigma(X \times B(X,Y)) \to B(X, \Sigma^{\star}(X+Y))$$

dinatural in $X \in \mathcal{C}$ and **natural** in $Y \in \mathcal{C}$.

A higher-order format for combinatory logic

Definition ($\mathcal{HO}$ rules)

$$\frac{(x_j \to y_j)_{j \in W} \quad (x_i \xrightarrow{z} y_i^z)_{i \in \{1,\ldots,n\} \setminus W,\, z \in \{x_1,\ldots,x_n\}}}{\mathsf{f}(x_1, \ldots, x_n) \to t}$$

$$\frac{(x_j \to y_j)_{j \in W} \quad (x_i \xrightarrow{z} y_i^z)_{i \in \{1,\ldots,n\} \setminus W,\, z \in \{x,x_1,\ldots,x_n\}}}{\mathsf{f}(x_1, \ldots, x_n) \xrightarrow{x} t}$$

A higher-order format for combinatory logic

Definition ($\mathcal{HO}$ rules)

$$\frac{(x_j \to y_j)_{j \in W} \quad (x_i \xrightarrow{z} y_i^z)_{i \in \{1,\ldots,n\} \setminus W,\, z \in \{x_1,\ldots,x_n\}}}{\mathsf{f}(x_1, \ldots, x_n) \to t}$$

$$\frac{(x_j \to y_j)_{j \in W} \quad (x_i \xrightarrow{z} y_i^z)_{i \in \{1,\ldots,n\} \setminus W,\, z \in \{x, x_1,\ldots,x_n\}}}{\mathsf{f}(x_1, \ldots, x_n) \xrightarrow{x} t}$$

Example rules (sugared)

$$\frac{}{S''(p,q) \xrightarrow{t} (p\,t)\,(q\,t)} \qquad \frac{p \to p'}{p\,q \to p'\,q} \qquad \frac{p \xrightarrow{q} p'}{p\,q \to p'}$$

Higher-Order Mathematical Operational Semantics

Proposition

For every finitary signature $\overline{\Sigma}$, with associated endofunctor $\Sigma\colon \mathsf{Set} \to \mathsf{Set}$, $\mathcal{HO}$ specifications are in a bijective correspondence with higher-order GSOS laws of Σ over $B(X, Y) = Y + Y^X$.

$$\frac{p \xrightarrow{q} p'}{p \quad q \longrightarrow p'}$$

$$\cong$$

$$\rho_X \colon \coprod_{\mathsf{f} \in \overline{\Sigma}} (\, X \;\times (\, Y \;+\; Y^{\;X}))^{\mathsf{ar}(\mathsf{f})} \to \Sigma^*(\, X \;+\; Y\,)$$

Higher-Order Mathematical Operational Semantics

Proposition

For every finitary signature $\overline{\Sigma}$, with associated endofunctor $\Sigma\colon \mathsf{Set} \to \mathsf{Set}$, $\mathcal{HO}$ specifications are in a bijective correspondence with higher-order GSOS laws of Σ over $B(X,Y) = Y + Y^X$.

$$\frac{\boxed{p} \xrightarrow{q} p'}{\boxed{p}\ \boxed{q} \longrightarrow p'}$$

$$\cong$$

$$\rho_X \colon \coprod_{\mathsf{f} \in \overline{\Sigma}} (\boxed{X} \times (Y + Y^X))^{\mathsf{ar}(\mathsf{f})} \to \Sigma^*(X + Y)$$

Higher-Order Mathematical Operational Semantics

Proposition

For every finitary signature $\overline{\Sigma}$, with associated endofunctor $\Sigma\colon \mathrm{Set} \to \mathrm{Set}$, $\mathcal{HO}$ specifications are in a bijective correspondence with higher-order GSOS laws of Σ over $B(X,Y) = Y + Y^X$.

$$\frac{\boxed{p} \xrightarrow{q} \boxed{p'}}{\boxed{p}\,\boxed{q} \longrightarrow p'}$$

$$\cong$$

$$\rho_X \colon \coprod_{\mathsf{f} \in \overline{\Sigma}} (\boxed{X} \times (\boxed{Y} + \boxed{Y}^X))^{\mathsf{ar}(\mathsf{f})} \to \Sigma^*(X + Y)$$

Proposition

For every finitary signature $\overline{\Sigma}$, with associated endofunctor $\Sigma\colon \mathrm{Set} \to \mathrm{Set}$, $\mathcal{HO}$ specifications are in a bijective correspondence with higher-order GSOS laws of Σ over $B(X, Y) = Y + Y^X$.

$$\frac{\boxed{p} \xrightarrow{q} \boxed{p'}}{\boxed{p}\ \boxed{q} \longrightarrow p'}$$

$$\cong$$

$$\rho_X \colon \coprod_{\mathsf{f} \in \overline{\Sigma}} (\boxed{X} \times (\boxed{Y} + \boxed{Y}^X))^{\mathsf{ar(f)}} \to \Sigma^*(\boxed{X} + \boxed{Y})$$

Higher-Order Mathematical Operational Semantics

Proposition

For every finitary signature $\overline{\Sigma}$, with associated endofunctor $\Sigma\colon \mathsf{Set} \to \mathsf{Set}$, $\mathcal{HO}$ specifications are in a bijective correspondence with higher-order GSOS laws of Σ over $B(X, Y) = Y + Y^X$.

$$\frac{\boxed{p} \xrightarrow{q} \boxed{p'}}{\boxed{p}\ \boxed{q} \to \boxed{p'}}$$

$$\cong$$

$$\rho_X\colon \coprod_{\mathsf{f}\in\overline{\Sigma}} (\boxed{X} \times (\boxed{Y} + \boxed{Y}^X))^{\mathsf{ar(f)}} \to \Sigma^*(\boxed{X} + \boxed{Y})$$

Higher-Order Abstract GSOS

Operational rules

$$\frac{p \xrightarrow{q} p'}{p\,q \to p'}$$

$$\frac{}{(\lambda x.p)\,q \to p[q/x]}$$

$\cong^\star$

Higher-order GSOS laws: (di-)natural trf.

$$\rho_{X,Y} \colon \underbrace{\Sigma(X \times B(X,Y))}_{\text{premises}} \to \underbrace{B(X, \Sigma^*(X+Y))}_{\text{conclusion}}$$

Higher-Order Abstract GSOS

Operational rules

$$\dfrac{\dfrac{p \xrightarrow{q} p'}{p\,q \to p'}}{(\lambda x.p)\; q \to p[q/x]}$$

$\cong^\star$

Higher-order GSOS laws: (di-)natural trf.

$$\rho_{X,Y} \colon \underbrace{\Sigma(X \times B(X,Y))}_{\text{premises}} \to \underbrace{B(X, \Sigma^*(X+Y))}_{\text{conclusion}}$$

For combinator calculi, we have

$$\mathcal{C} = \mathrm{Set}$$

$$\Sigma X = 1 + X \times X + \dots$$

$$B(X,Y) = Y + Y^X$$

β-reduction or combinator

Higher-Order Abstract GSOS

Operational rules

$$\frac{p \xrightarrow{q} p'}{p\,q \to p'}$$

$$\frac{}{(\lambda x.p)\,q \to p[q/x]}$$

$\cong^\star$

Higher-order GSOS laws: (di-)natural trf.

$$\rho_{X,Y} \colon \underbrace{\Sigma(X \times B(X,Y))}_{\text{premises}} \to \underbrace{B(X, \Sigma^*(X+Y))}_{\text{conclusion}}$$

For the call-by-name λ-calculus, we have

$$\mathcal{C} = \mathrm{Set}^{\mathbb{F}}$$

$$\Sigma X = V + \delta X + X \times X \quad \text{(Fiore, Plotkin and Turi [15])}$$

$$B(X,Y) = \langle X, Y \rangle \times (Y + Y^X + 1)$$

substitution stucture

β-reduction, λ-expr or stuck

Higher-Order Abstract GSOS

Operational rules

$$\frac{}{(\lambda x.p)\, q \to p[q/x]}$$

$$\frac{p \to p'}{p\, q \to p'\, q}$$

$\cong^{\star}$

Higher-order GSOS laws: (di-)natural trf.

$$\rho_{X,Y} \colon \underbrace{\Sigma(X \times B(X, Y))}_{\text{premises}} \to \underbrace{B(X, \Sigma^*(X + Y))}_{\text{conclusion}}$$

Higher-Order Abstract GSOS

Operational rules

$$\frac{}{(\lambda x.p)\, q \to p[q/x]}$$

$$\frac{p \to p'}{p\, q \to p'\, q}$$

$\cong^\star$

Higher-order GSOS laws: (di-)natural trf.

$$\rho_{X,Y} : \underbrace{\Sigma(X \times B(X,Y))}_{\text{premises}} \to \underbrace{B(X, \Sigma^*(X+Y))}_{\text{conclusion}}$$

- Operational model $\gamma : \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$, ~~denotational model~~.

  programs

  e.g. $\gamma(t) = t'$ if $t \to t'$ and $\gamma(\lambda x.M) = (e \mapsto M[e/x])$, $(\gamma(I) = \mathrm{id}$ for SKI$)$

Higher-Order Abstract GSOS

Operational rules

$$\frac{}{(\lambda x.p)\, q \to p[q/x]}$$

$$\frac{p \to p'}{p\, q \to p'\, q}$$

$\cong^\star$

Higher-order GSOS laws: (di-)natural trf.

$$\rho_{X,Y} : \underbrace{\Sigma(X \times B(X,Y))}_{\text{premises}} \to \underbrace{B(X, \Sigma^*(X+Y))}_{\text{conclusion}}$$

- Operational model $\gamma : \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$, ~~denotational model~~.

  programs

  e.g. $\gamma(t) = t'$ if $t \to t'$ and $\gamma(\lambda x.M) = (e \mapsto M[e/x])$, $(\gamma(I) = \mathrm{id}$ for SKI$)$

- **Key feature:** compositionality, i.e. bisimilarity is a congruence.

  Proof: more complex than first-order case + needs mild assumptions.

Strong Applicative Bisimilarity

Coalgebraic bisimilarity on operational model $\mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$

$$=$$

strong applicative bisimilarity.

Strong Applicative Bisimilarity

Coalgebraic bisimilarity on operational model $\mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$

$=$

strong applicative bisimilarity.

Example: λ-calculus

closed λ-terms

Greatest relation $\sim \subseteq \Lambda \times \Lambda$ such that for $t_1 \sim t_2$,

$$t_1 \to t_1' \implies t_2 \to t_2' \quad \wedge \quad t_1' \sim t_2';$$
$$t_1 = \lambda x.t_1' \implies t_2 = \lambda x.t_2' \wedge \forall e \in \Lambda.\ t_1'[e/x] \sim t_2'[e/x];$$

$+$ two symmetric conditions

Abstract odelling of Operational Semantics

Concrete/Abstract

Abstract odelling of Operational Semantics

Concrete/Abstract

1. Algebraic signature Σ

Abstract odelling of Operational Semantics

Concrete/Abstract

1. Algebraic signature Σ

1. Syntax endofunctor $\Sigma\colon \mathcal{C} \to \mathcal{C}$

Abstract odelling of Operational Semantics

Concrete/Abstract

1. Algebraic signature Σ
2. Program terms $\mu\Sigma$

1. Syntax endofunctor $\Sigma\colon \mathcal{C} \to \mathcal{C}$

Abstract odelling of Operational Semantics

Concrete/Abstract

Concrete	Abstract
1. Algebraic signature Σ	1. Syntax endofunctor $\Sigma\colon \mathcal{C} \to \mathcal{C}$
2. Program terms $\mu\Sigma$	2. Initial Σ-algebra $\mu\Sigma = \Sigma^*(0)$

Abstract odelling of Operational Semantics

Concrete/Abstract

1. Algebraic signature Σ
2. Program terms $\mu\Sigma$
3. (Impl.) nature of computation

1. Syntax endofunctor $\Sigma\colon \mathcal{C} \to \mathcal{C}$
2. Initial Σ-algebra $\mu\Sigma = \Sigma^*(0)$

Abstract odelling of Operational Semantics

Concrete/Abstract

1. Algebraic signature Σ
2. Program terms $\mu\Sigma$
3. (Impl.) nature of computation

1. Syntax endofunctor $\Sigma\colon \mathcal{C} \to \mathcal{C}$
2. Initial Σ-algebra $\mu\Sigma = \Sigma^*(0)$
3. Bifunctor $B\colon \mathcal{C}^{\mathsf{op}} \times \mathcal{C} \to \mathcal{C}$

Abstract odelling of Operational Semantics

Concrete/Abstract

1. Algebraic signature Σ
2. Program terms $\mu\Sigma$
3. (Impl.) nature of computation
4. Operational rules $\dfrac{t \to t'}{t \cdot s \to t' \cdot s}$

1. Syntax endofunctor $\Sigma\colon \mathcal{C} \to \mathcal{C}$
2. Initial Σ-algebra $\mu\Sigma = \Sigma^*(0)$
3. Bifunctor $B\colon \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathcal{C}$

Abstract odelling of Operational Semantics

Concrete/Abstract

1. Algebraic signature Σ
2. Program terms $\mu\Sigma$
3. (Impl.) nature of computation
4. Operational rules $\frac{t \to t'}{t \cdot s \to t' \cdot s}$

1. Syntax endofunctor $\Sigma \colon \mathcal{C} \to \mathcal{C}$
2. Initial Σ-algebra $\mu\Sigma = \Sigma^*(0)$
3. Bifunctor $B \colon \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathcal{C}$
4. Higher-order GSOS law $\rho_{X,Y}$

Abstract odelling of Operational Semantics

Concrete/Abstract

1. Algebraic signature Σ
2. Program terms $\mu\Sigma$
3. (Impl.) nature of computation
4. Operational rules $\frac{t \to t'}{t \cdot s \to t' \cdot s}$
5. Oper. model $t \to t'$, $t, t' \in \mu\Sigma$

1. Syntax endofunctor $\Sigma\colon \mathcal{C} \to \mathcal{C}$
2. Initial Σ-algebra $\mu\Sigma = \Sigma^*(0)$
3. Bifunctor $B\colon \mathcal{C}^{\mathsf{op}} \times \mathcal{C} \to \mathcal{C}$
4. Higher-order GSOS law $\rho_{X,Y}$

Abstract odelling of Operational Semantics

Concrete/Abstract

1. Algebraic signature Σ
2. Program terms $\mu\Sigma$
3. (Impl.) nature of computation
4. Operational rules $\dfrac{t \to t'}{t \cdot s \to t' \cdot s}$
5. Oper. model $t \to t'$, $t, t' \in \mu\Sigma$

1. Syntax endofunctor $\Sigma\colon \mathcal{C} \to \mathcal{C}$
2. Initial Σ-algebra $\mu\Sigma = \Sigma^*(0)$
3. Bifunctor $B\colon \mathcal{C}^{\mathsf{op}} \times \mathcal{C} \to \mathcal{C}$
4. Higher-order GSOS law $\rho_{X,Y}$
5. Coalgebra $\gamma\colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$

Abstract odelling of Operational Semantics

Concrete/Abstract

1. Algebraic signature Σ
2. Program terms $\mu\Sigma$
3. (Impl.) nature of computation
4. Operational rules $\frac{t \to t'}{t \cdot s \to t' \cdot s}$
5. Oper. model $t \to t'$, $t, t' \in \mu\Sigma$
6. Strong applicative bisimulation

1. Syntax endofunctor $\Sigma\colon \mathcal{C} \to \mathcal{C}$
2. Initial Σ-algebra $\mu\Sigma = \Sigma^*(0)$
3. Bifunctor $B\colon \mathcal{C}^{\mathsf{op}} \times \mathcal{C} \to \mathcal{C}$
4. Higher-order GSOS law $\rho_{X,Y}$
5. Coalgebra $\gamma\colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$

Abstract odelling of Operational Semantics

Concrete/Abstract

Concrete	Abstract
1. Algebraic signature Σ	1. Syntax endofunctor $\Sigma\colon \mathcal{C} \to \mathcal{C}$
2. Program terms $\mu\Sigma$	2. Initial Σ-algebra $\mu\Sigma = \Sigma^*(0)$
3. (Impl.) nature of computation	3. Bifunctor $B\colon \mathcal{C}^{\mathsf{op}} \times \mathcal{C} \to \mathcal{C}$
4. Operational rules $\frac{t \to t'}{t \cdot s \to t' \cdot s}$	4. Higher-order GSOS law $\rho_{X,Y}$
5. Oper. model $t \to t'$, $t, t' \in \mu\Sigma$	5. Coalgebra $\gamma\colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$
6. Strong applicative bisimulation	6. $B(\mu\Sigma, -)$-bisimulations

Abstract odelling of Operational Semantics

Concrete/Abstract

Concrete	Abstract
1. Algebraic signature Σ	1. Syntax endofunctor $\Sigma\colon \mathcal{C} \to \mathcal{C}$
2. Program terms $\mu\Sigma$	2. Initial Σ-algebra $\mu\Sigma = \Sigma^*(0)$
3. (Impl.) nature of computation	3. Bifunctor $B\colon \mathcal{C}^{\mathsf{op}} \times \mathcal{C} \to \mathcal{C}$
4. Operational rules $\frac{t \to t'}{t \cdot s \to t' \cdot s}$	4. Higher-order GSOS law $\rho_{X,Y}$
5. Oper. model $t \to t'$, $t, t' \in \mu\Sigma$	5. Coalgebra $\gamma\colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$
6. Strong applicative bisimulation	6. $B(\mu\Sigma, -)$-bisimulations

Assuming a suitable category $\mathcal{C}$.

Abstract odelling of Operational Semantics

Concrete/Abstract

1. Algebraic signature Σ
2. Program terms $\mu\Sigma$
3. (Impl.) nature of computation
4. Operational rules $\frac{t \to t'}{t \cdot s \to t' \cdot s}$
5. Oper. model $t \to t'$, $t, t' \in \mu\Sigma$
6. Strong applicative bisimulation

1. Syntax endofunctor $\Sigma \colon \mathcal{C} \to \mathcal{C}$
2. Initial Σ-algebra $\mu\Sigma = \Sigma^*(0)$
3. Bifunctor $B \colon \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathcal{C}$
4. Higher-order GSOS law $\rho_{X,Y}$
5. Coalgebra $\gamma \colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$
6. $B(\mu\Sigma, -)$-bisimulations

Assuming a suitable category $\mathcal{C}$.

[16]: Congruence of bisimilarity, for free!

Question marks

Concrete/Abstract

Question marks

Concrete/Abstract

8. Howe's closure
9. Howe's method
10. Logical predicates/relations
11. Fundamental Properties

Question marks

Concrete	Abstract
8. Howe's closure	8. ???
9. Howe's method	9. ???
10. Logical predicates/relations	10. ???
11. Fundamental Properties	11. ???

Question marks

Concrete/Abstract

8. Howe's closure	8. ???
9. Howe's method	9. ???
10. Logical predicates/relations	10. ???
11. Fundamental Properties	11. ???

We want to model all of the above **generically**, in a **language-independent** manner.

Question marks

Concrete/Abstract

8. Howe's closure	8. ???
9. Howe's method	9. ???
10. Logical predicates/relations	10. ???
11. Fundamental Properties	11. ???

We want to model ove **generically**, in **ndependent** manner.

Relation Lifting!

Predicate Lifting!

Relational Reasoning

How to do program discourse, categorically

Key concept 1: If $\mathcal{C}$ is our base universe of discourse, we can form the categories $\mathrm{Rel}(\mathcal{C})$ and $\mathrm{Pred}(\mathcal{C})$ of resp. (homogenous) relations and predicates on $\mathcal{C}$. These are the categories of subobjects on rep. $X \times X$ and X.

$$\begin{tikzcd}
R \arrow[r, dashed] \arrow[d, tail, "\langle \mathsf{l}_R, \mathsf{r}_R \rangle"'] & S \arrow[d, tail, "\langle \mathsf{l}_S, \mathsf{r}_S \rangle"] \\
X \times X \arrow[r, "f \times f"] & Y \times Y
\end{tikzcd}
\qquad
\begin{tikzcd}
P \arrow[r, dashed] \arrow[d, tail, "p"'] & Q \arrow[d, tail, "q"] \\
X \arrow[r, "f"] & Y
\end{tikzcd}$$

How to do program discourse, categorically

Key concept 1: If $\mathcal{C}$ is our base universe of discourse, we can form the categories $\mathsf{Rel}(\mathcal{C})$ and $\mathsf{Pred}(\mathcal{C})$ of resp. (homogenous) relations and predicates on $\mathcal{C}$. These are the categories of subobjects on rep. $X \times X$ and X.

$$
\begin{tikzcd}
R \arrow[r, dashed] \arrow[d, tail, "\langle l_R, r_R \rangle"'] & S \arrow[d, tail, "\langle l_S, r_S \rangle"] \\
X \times X \arrow[r, "f \times f"] & Y \times Y
\end{tikzcd}
\qquad
\begin{tikzcd}
P \arrow[r, dashed] \arrow[d, tail, "p"'] & Q \arrow[d, tail, "q"] \\
X \arrow[r, "f"] & Y
\end{tikzcd}
$$

Key concept 2: We extend the functors to $\mathsf{Rel}(\mathcal{C})$ and $\mathsf{Pred}(\mathcal{C})$, a process that is known as relation (or predicate) lifting [17].

$$
\begin{tikzcd}
\mathsf{Rel}(\mathcal{C}) \arrow[r, "\overline{\Sigma}"] \arrow[d, "|-|"'] & \mathsf{Rel}(\mathcal{C}) \arrow[d, "|-|"] \\
\mathcal{C} \arrow[r, "\Sigma"] & \mathcal{C}
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathsf{Pred}(\mathcal{C}) \arrow[r, "\overline{\Sigma}"] \arrow[d, "|-|"'] & \mathsf{Pred}(\mathcal{C}) \arrow[d, "|-|"] \\
\mathcal{C} \arrow[r, "\Sigma"] & \mathcal{C}
\end{tikzcd}
$$

How to do program discourse, categorically

Key concept 1: If $\mathcal{C}$ is our base universe of discourse, we can form the categories $\mathsf{Rel}(\mathcal{C})$ and $\mathsf{Pred}(\mathcal{C})$ of resp. (homogenous) relations and predicates on $\mathcal{C}$. These are the categories of subobjects on rep. $X \times X$ and X.

$$\begin{tikzcd} R \arrow[r, dashed] \arrow[d, tail, "\langle \mathsf{l}_R, \mathsf{r}_R \rangle"'] & S \arrow[d, tail, "\langle \mathsf{l}_S, \mathsf{r}_S \rangle"] \\ X \times X \arrow[r, "f \times f"] & Y \times Y \end{tikzcd} \qquad \begin{tikzcd} P \arrow[r, dashed] \arrow[d, tail, "p"'] & Q \arrow[d, tail, "q"] \\ X \arrow[r, "f"] & Y \end{tikzcd}$$

Key concept 2: We extend the functors to $\mathsf{Rel}(\mathcal{C})$ and $\mathsf{Pred}(\mathcal{C})$, a process that is known as relation (or predicate) lifting [17].

$$\begin{tikzcd} \mathsf{Rel}(\mathcal{C}) \arrow[r, "\overline{\Sigma}"] \arrow[d, "|-|"'] & \mathsf{Rel}(\mathcal{C}) \arrow[d, "|-|"] \\ \mathcal{C} \arrow[r, "\Sigma"] & \mathcal{C} \end{tikzcd} \qquad \begin{tikzcd} \mathsf{Pred}(\mathcal{C}) \arrow[r, "\overline{\Sigma}"] \arrow[d, "|-|"'] & \mathsf{Pred}(\mathcal{C}) \arrow[d, "|-|"] \\ \mathcal{C} \arrow[r, "\Sigma"] & \mathcal{C} \end{tikzcd}$$

Also, write $\mathsf{Pred}_X(\mathcal{C})$, $\mathsf{Rel}_X(\mathcal{C})$ for the lattices of resp. predicates and relations on X.

Act I, Induction. Part 1, Predicates.

Let $P \rightarrowtail \mu\Sigma$ be a predicate on terms (assume a typed syntax, for the heck of it).

Structural induction

1. (Repeat for every operation) For all $t : \tau_1 \multimap \tau_2$, $s : \tau_1$ such that $P_{\tau_1 \multimap \tau_2}(t)$ and $P_{\tau_1}(s)$, then $P_{\tau_2}(t\, s)$.
2. By induction, for all types τ and terms $t : \tau$, $P_\tau(t)$.

Unary induction proof principle

1. $\overline{\Sigma}(P)$ represents 1-depth terms (operations) whose subterms are in P ($\overline{\Sigma}$ is the canonical lifting). There is a Σ-algebra structure

$$\overline{\Sigma}(P) \leq \iota^\star[P], \text{ where } \iota \colon \Sigma\mu\Sigma \to \mu\Sigma \text{ is the initial } \Sigma\text{-algebra.}$$

2. As initial algebras have no proper subalgebras, $P \cong \mu\Sigma$.

Act I, Induction. Part 2, Relations.

Let $R \rightarrowtail \mu\Sigma \times \mu\Sigma$ be a relation on terms.

Structural induction (Fundamental Property)

1. (Repeat for every operation) For all $t_1, t_2 : \tau_1 \rightharpoonup \tau_2$, $s_1, s_2 : \tau_1$ such that $R_{\tau_1 \rightharpoonup \tau_2}(t_1, t_2)$ and $R_{\tau_1}(s_1, s_2)$, then $R_{\tau_2}(t_2\, s_2, t_2\, s_2)$.
2. Then for all types τ, relation R_τ is reflexive.

Binary induction proof principle

1. $\overline{\Sigma}(R)$ represents pairs of 1-depth terms with subterms in R. If there is

$$\overline{\Sigma}(R) \leq (\iota \times \iota)^\star[R] \text{(that is, } R \text{ is a congruence),}$$

2. then $\Delta \leq R$ because all congruences on an initial algebra are reflexive.

Act II, Bisimulations. Prelude.

Simple go-to example (untyped syntax this time)

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

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

Act II, Bisimulations. Prelude.

Simple go-to example (untyped syntax this time)

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

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

$$\begin{tikzcd}
\mathsf{Pred}(\mathcal{C})^{\mathsf{op}} \times \mathsf{Pred}(\mathcal{C}) \arrow[r, "\overline{B}"] \arrow[d, "|-|^{\mathsf{op}} \times |-|"'] & \mathsf{Pred}(\mathcal{C}) \arrow[d, "|-|"] \\
\mathcal{C}^{\mathsf{op}} \times \mathcal{C} \arrow[r, "B"] & \mathcal{C}
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathsf{Rel}(\mathcal{C})^{\mathsf{op}} \times \mathsf{Rel}(\mathcal{C}) \arrow[r, "\overline{B}"] \arrow[d, "|-|^{\mathsf{op}} \times |-|"'] & \mathsf{Rel}(\mathcal{C}) \arrow[d, "|-|"] \\
\mathcal{C}^{\mathsf{op}} \times \mathcal{C} \arrow[r, "B"] & \mathcal{C}
\end{tikzcd}$$

Simple go-to example (untyped syntax this time)

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

$$\begin{tikzcd}
\mathrm{Pred}(\mathcal{C})^{\mathsf{op}} \times \mathrm{Pred}(\mathcal{C}) \arrow[r, "\overline{B}"] \arrow[d, "|-|^{\mathsf{op}} \times |-|"'] & \mathrm{Pred}(\mathcal{C}) \arrow[d, "|-|"] \\
\mathcal{C}^{\mathsf{op}} \times \mathcal{C} \arrow[r, "B"] & \mathcal{C}
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathrm{Rel}(\mathcal{C})^{\mathsf{op}} \times \mathrm{Rel}(\mathcal{C}) \arrow[r, "\overline{B}"] \arrow[d, "|-|^{\mathsf{op}} \times |-|"'] & \mathrm{Rel}(\mathcal{C}) \arrow[d, "|-|"] \\
\mathcal{C}^{\mathsf{op}} \times \mathcal{C} \arrow[r, "B"] & \mathcal{C}
\end{tikzcd}$$

Let $P, Q \subseteq \mu\Sigma$ be predicates. Then $\overline{B}(P,Q) \subseteq \mu\Sigma + \mu\Sigma^{\mu\Sigma}$ amounts to the following:

$$\overline{B}(P,Q) = \{t \mid Q(t)\} \vee \{f \in \mu\Sigma^{\mu\Sigma} \mid \forall t.\, P(t) \implies Q(f(t))\},$$

aka, inputs in P are mapped to outputs in Q!

Simple go-to example (untyped syntax this time)

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

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

$$\begin{tikzcd}
\mathrm{Pred}(\mathcal{C})^{\mathrm{op}} \times \mathrm{Pred}(\mathcal{C}) \arrow[r, "\overline{B}"] \arrow[d, "|-|^{\mathrm{op}} \times |-|"'] & \mathrm{Pred}(\mathcal{C}) \arrow[d, "|-|"] \\
\mathcal{C}^{\mathrm{op}} \times \mathcal{C} \arrow[r, "B"] & \mathcal{C}
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathrm{Rel}(\mathcal{C})^{\mathrm{op}} \times \mathrm{Rel}(\mathcal{C}) \arrow[r, "\overline{B}"] \arrow[d, "|-|^{\mathrm{op}} \times |-|"'] & \mathrm{Rel}(\mathcal{C}) \arrow[d, "|-|"] \\
\mathcal{C}^{\mathrm{op}} \times \mathcal{C} \arrow[r, "B"] & \mathcal{C}
\end{tikzcd}$$

Let $R, S \subseteq \mu\Sigma \times \mu\Sigma$ be relations. Then $\overline{B}(R,S)$ amounts to the following:

$$\overline{B}(R,S) = \{(t_1,t_2) \mid Q(t_1,t_2)\} \vee \{f \in \mu\Sigma^{\mu\Sigma} \mid \forall t_1, t_2.\, R(t_1,t_2) \implies Q(f(t_1), f(t_2))\},$$

aka, related inputs are mapped to related outputs!

Act II, Bisimulations. Part 1, Predicates.

Let $P, Q \rightarrowtail X$ be predicates on the state space of a coalgebra $h : X \to B(X, X)$. We say that P is a (Q-relative) ($\overline{B}$)-invariant [18] if

$$P \leq h^{\star}[\overline{B}(Q, P)]$$

We say that an invariant P is **logical** if it is relative to itself.

Instantiate on $\gamma : \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$. A predicate P is logical if the following hold:

1. If $t = \lambda x.s$, then for all e with $P(e)$, then $P(s[e/x])$.
2. If $t \to t'$ then $P(t')$.

The above notion instantiates correctly in other settings (assuming the coalgebra is setup correctly), e.g. typed: the tuple $(t, s) : \tau_1 \times \tau_2$ is in P when $P_{\tau_1}(t)$ and $P_{\tau_2}(s)$.

Bisimulations, logical relations and step-indexing [19]

Let $h : X \to B(X, X)$ be a coalgebra and $\tilde{h} : X \to B(X, X)$ be a weakening of h (think $\to$ vs its saturation/closure $\Rightarrow$). We say that:

1. A relation $R \rightarrowtail X \times X$ is a **bisimulation** if $R \leq (h \times \tilde{h})^{\star}[\overline{B}(\Delta, R)]$.
2. A relation $R \rightarrowtail X \times X$ is a **logical relation** if $R \leq (h \times \tilde{h})^{\star}[\overline{B}(R, R)]$.
3. An ordinal-indexed family of relations $(R^{\alpha} \rightarrowtail X \times X)_{\alpha}$ is a **step-indexed logical relation** if it forms a decreasing chain (i.e. $R^{\alpha} \leq R^{\beta}$ for all $\beta < \alpha$) and satisfies

$$R^{\alpha+1} \leq (h \times \tilde{h})^{\star}[\overline{B}(R^{\alpha}, R^{\alpha})] \quad \text{for all } \alpha.$$

Bisimulations, logical relations and step-indexing [19]

Let $h : X \to B(X, X)$ be a coalgebra and $\tilde{h} : X \to B(X, X)$ be a weakening of h (think $\to$ vs its saturation/closure $\Rightarrow$). We say that:

1. A relation R on X is a **($\overline{B}$-)bisimulation** (for $h, \tilde{h}$) if $R \leq (h \times \tilde{h})^\star[\overline{B}(\Delta, R)]$.
2. A relation R on X is a **($\overline{B}$-)logical relation** (for $h, \tilde{h}$) if $R \leq (h \times \tilde{h})^\star[\overline{B}(R, R)]$.
3. An ordinal-indexed family of relations $(R^\alpha \rightarrowtail X \times X)_\alpha$ is a **($\overline{B}$-)step-indexed logical relation** (for $h, \tilde{h}$) if it forms a decreasing chain (i.e. $R^\alpha \leq R^\beta$ for all $\beta < \alpha$) and satisfies

$$R^{\alpha+1} \leq (h \times \tilde{h})^\star[\overline{B}(R^\alpha, R^\alpha)] \quad \text{for all } \alpha.$$

Simple example

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

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

We will use the asymmetric Egli-Milner relation lifting for $\mathcal{P}B, \widetilde{\mathcal{P}B}$.

Simple example

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

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

We will use the asymmetric Egli-Milner relation lifting for $\mathcal{P}B, \widetilde{\mathcal{P}B}$.

Notation (reminder and introduction)

- Write $t \xrightarrow{e} t'$ if $t = \lambda x.M$ and $t' = M[e/x] = \gamma(t)(e)$.
- Write $t \Rightarrow t'$ if $t \to t_1 \to \cdots \to t_n \to t'$.
- Write $t \overset{e}{\Rightarrow} t'$ if $t \to t_1 \to \cdots \to t_n \to t''$ and $t'' \xrightarrow{e} t'$.
- The system $\Rightarrow$ is modelled by $\tilde{\gamma} : \mu\Sigma \to \mathcal{P}(B(\mu\Sigma, \mu\Sigma))$ (technically $\Rightarrow$ is a notation for $\tilde{\gamma}$).

Simple example

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

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

We will use the asymmetric Egli-Milner relation lifting for $\mathcal{P}B, \widetilde{\mathcal{P}B}$.

Let $\tilde{\gamma}$ be the closure of γ under β reductions. A relation $R \subseteq \mu\Sigma \times \mu\Sigma$ is a $(\widetilde{\mathcal{P}B})$-bisimulation (for $\gamma, \tilde{\gamma}$) if for all t, s with $R(t,s)$, the following are true:

- If $t \to t'$ then $s \Rightarrow s'$ and $R(t', s')$.
- For all e, if $t \xrightarrow{e} t'$, then $s \xRightarrow{e} s'$ and $R(t', s')$.

Applicative simulation!

Simple example

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

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

We will use the asymmetric Egli-Milner relation lifting for $\mathcal{P}B, \widetilde{\mathcal{P}B}$.

Simple example

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

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

We will use the asymmetric Egli-Milner relation lifting for $\mathcal{P}B, \widetilde{\mathcal{P}B}$.

Let $\tilde{\gamma}$ be the closure of γ under β reductions. A relation $R \subseteq \mu\Sigma \times \mu\Sigma$ is a $(\widetilde{\mathcal{P}B})$-logical relation (for $\gamma, \tilde{\gamma}$) if for all t, s with $R(t,s)$, the following are true:

- If $t \to t'$ then $s \Rightarrow s'$ and $R(t', s')$.
- For all e_1, e_2 with $R(e_1, e_2)$, if $t \xrightarrow{e_1} t'$, then $s \xRightarrow{e_2} s'$ and $R(t', s')$.

Act II, Bisimulations. Part 2, Relations.

Simple example

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

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

We will use the asymmetric Egli-Milner relation lifting for $\mathcal{P}B, \widetilde{\mathcal{P}B}$.

Let $\tilde{\gamma}$ be the closure of γ under β reductions. A relation $R \subseteq \mu\Sigma \times \mu\Sigma$ is a $(\widetilde{\mathcal{P}B})$-logical relation (for $\gamma, \tilde{\gamma}$) if for all t, s with $R(t,s)$, the following are true:

- If $t \to t'$ then $s \Rightarrow s'$ and $R(t', s')$.
- For all e_1, e_2 with $R(e_1, e_2)$, if $t \xrightarrow{e_1} t'$, then $s \xRightarrow{e_2} s'$ and $R(t', s')$.

Logical preorder! Kind of concurrent flavor when $\to, \Rightarrow$ is used instead of $\Downarrow, \Downarrow$.

Simple example

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

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

We will use the asymmetric Egli-Milner relation lifting for $\mathcal{P}B, \widetilde{\mathcal{P}B}$.

Act II, Bisimulations. Part 2, Relations.

Simple example

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

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

We will use the asymmetric Egli-Milner relation lifting for $\mathcal{P}B, \widetilde{\mathcal{P}B}$.

Let $\tilde{\gamma}$ be the closure of γ under β reductions. A family $(R^\alpha \subseteq \mu\Sigma \times \mu\Sigma)_\alpha$ is a step-indexed $(\widetilde{\mathcal{P}B})$-logical relation (for $\gamma, \tilde{\gamma}$) if $\forall \alpha, \beta$ with $\beta < \alpha$, $R^\alpha \leq R^\beta$ and for all α, t, s with $R^{\alpha+1}(t,s)$, the following are true:

- If $t \to t'$ then $s \Rightarrow s'$ and $R^\alpha(t', s')$.
- For all e_1, e_2 with $R^\alpha(e_1, e_2)$, if $t \xrightarrow{e_1} t'$, then $s \xRightarrow{e_2} s'$ and $R^\alpha(t', s')$.

This was supposed to be an example on a typed λ-calculus, but I ran out of time while preparing the slides. We can do it on the board, depending on time.

Abstract modelling of Predicates and Relations

Concrete/Abstract

Abstract modelling of Predicates and Relations

Concrete/Abstract

1. Predicates, relations on terms

Abstract modelling of Predicates and Relations

Concrete/Abstract

Concrete	Abstract
1. Predicates, relations on terms	1. $P \rightarrowtail \mu\Sigma$, $R \rightarrowtail \mu\Sigma \times \mu\Sigma$

Abstract modelling of Predicates and Relations

Concrete/Abstract

1. Predicates, relations on terms
2. Predicate, relational reasoning

1. $P \rightarrowtail \mu\Sigma$, $R \rightarrowtail \mu\Sigma \times \mu\Sigma$

Abstract modelling of Predicates and Relations

Concrete/Abstract

1. Predicates, relations on terms
2. Predicate, relational reasoning

1. $P \rightarrowtail \mu\Sigma$, $R \rightarrowtail \mu\Sigma \times \mu\Sigma$
2. Complete, well-powered cat. $\mathcal{C}$

Abstract modelling of Predicates and Relations

Concrete/Abstract

1. Predicates, relations on terms
2. Predicate, relational reasoning
3. (P is a) Logical Predicate

1. $P \rightarrowtail \mu\Sigma$, $R \rightarrowtail \mu\Sigma \times \mu\Sigma$
2. Complete, well-powered cat. $\mathcal{C}$

Abstract modelling of Predicates and Relations

Concrete/Abstract

Concrete	Abstract
1. Predicates, relations on terms	1. $P \rightarrowtail \mu\Sigma$, $R \rightarrowtail \mu\Sigma \times \mu\Sigma$
2. Predicate, relational reasoning	2. Complete, well-powered cat. $\mathcal{C}$
3. (P is a) Logical Predicate	3. $P \leq h^{\star}[\overline{B}(P, P)]$

Abstract modelling of Predicates and Relations

Concrete/Abstract

1. Predicates, relations on terms
2. Predicate, relational reasoning
3. (P is a) Logical Predicate
4. (R is a) Logical Relation

1. $P \rightarrowtail \mu\Sigma$, $R \rightarrowtail \mu\Sigma \times \mu\Sigma$
2. Complete, well-powered cat. $\mathcal{C}$
3. $P \leq h^{\star}[\overline{B}(P, P)]$

Abstract modelling of Predicates and Relations

Concrete/Abstract

Concrete	Abstract
1. Predicates, relations on terms	1. $P \rightarrowtail \mu\Sigma$, $R \rightarrowtail \mu\Sigma \times \mu\Sigma$
2. Predicate, relational reasoning	2. Complete, well-powered cat. $\mathcal{C}$
3. (P is a) Logical Predicate	3. $P \leq h^\star[\overline{B}(P, P)]$
4. (R is a) Logical Relation	4. $R \leq (h \times \tilde{h})^\star[\overline{B}(R, R)]$

Abstract modelling of Predicates and Relations

Concrete/Abstract

Concrete	Abstract
1. Predicates, relations on terms	1. $P \rightarrowtail \mu\Sigma$, $R \rightarrowtail \mu\Sigma \times \mu\Sigma$
2. Predicate, relational reasoning	2. Complete, well-powered cat. $\mathcal{C}$
3. (P is a) Logical Predicate	3. $P \leq h^\star[\overline{B}(P, P)]$
4. (R is a) Logical Relation	4. $R \leq (h \times \tilde{h})^\star[\overline{B}(R, R)]$
5. Fundamental Property of Logical Relations	

Abstract modelling of Predicates and Relations

Concrete/Abstract

Concrete	Abstract
1. Predicates, relations on terms	1. $P \rightarrowtail \mu\Sigma$, $R \rightarrowtail \mu\Sigma \times \mu\Sigma$
2. Predicate, relational reasoning	2. Complete, well-powered cat. $\mathcal{C}$
3. (P is a) Logical Predicate	3. $P \leq h^\star[\overline{B}(P, P)]$
4. (R is a) Logical Relation	4. $R \leq (h \times \tilde{h})^\star[\overline{B}(R, R)]$
5. Fundamental Property of Logical Relations	5. Generalized induction $\overline{\Sigma}(R) \leq \iota^\star[R] \implies \Delta \leq R$

Recall that relation lifting is algebraic and coalgebraic, and independent of the Higher-order Abstract GSOS framework.

However, the marriage of algebra and coalgebra that HO Abstract GSOS represents extends along their liftings :).

Howe's method in higher-order Abstract GSOS, briefly

Motivation: We need congruence of applicative similarity, not of its strong version.

[2] Howe's method is complex, clunky, conceptually mysterious and unclear why it works. Shout-out to people uncovering its mysteries (Dal Lago et al.[20], Borthelle et al. [21], Hirschowiz and Lafont [13]).

Howe's method in higher-order Abstract GSOS, briefly

Motivation: We need congruence of applicative similarity, not of its strong version.

Plan: Redo Howe's method using our abstract machinery, such that:

- We systematize it into a generic, language-independent method.
- Expose its core ideas, its language-specific part, and then simplify [2].

[2] Howe's method is complex, clunky, conceptually mysterious and unclear why it works. Shout-out to people uncovering its mysteries (Dal Lago et al.[20], Borthelle et al. [21], Hirschowiz and Lafont [13]).

Howe's method in higher-order Abstract GSOS, briefly

Motivation: We need congruence of applicative similarity, not of its strong version.

Plan: Redo Howe's method using our abstract machinery, such that:

- We systematize it into a generic, language-independent method.
- Expose its core ideas, its language-specific part, and then simplify [2].

Key concept: Howe's closure $\hat{R}$: initial algebra (lfp) of an endofunctor on $\mathrm{Rel}_{\mu\Sigma}(\mathcal{C})$!

$$\text{For an applicative simulation } R \rightarrowtail \mu\Sigma \times \mu\Sigma, \quad \hat{R} = \mu S.\, R \vee \iota_\star[\overline{\Sigma}(S)]; R.$$

[2] Howe's method is complex, clunky, conceptually mysterious and unclear why it works. Shout-out to people uncovering its mysteries (Dal Lago et al.[20], Borthelle et al. [21], Hirschowiz and Lafont [13]).

Howe's method in higher-order Abstract GSOS, briefly

Motivation: We need congruence of applicative similarity, not of its strong version.

Plan: Redo Howe's method using our abstract machinery, such that:

- We systematize it into a generic, language-independent method.
- Expose its core ideas, its language-specific part, and then simplify [2].

Key concept: Howe's closure $\hat{R}$: initial algebra (lfp) of an endofunctor on $\mathrm{Rel}_{\mu\Sigma}(\mathcal{C})$!

$$\text{For an applicative simulation } R \rightarrowtail \mu\Sigma \times \mu\Sigma, \quad \hat{R} = \mu S.\, R \vee \iota_\star[\overline{\Sigma}(S)]; R.$$

Results [22]: For $\hat{R}$ to be a bisimulation, just show that "weakened" rules are sound:

$$\frac{t \overset{s}{\Rightarrow} t'}{t\, s \Rightarrow t'} \quad \checkmark \qquad\qquad \frac{t \Rightarrow t'}{t\, s \Rightarrow t'\, s} \quad \checkmark$$

[2] Howe's method is complex, clunky, conceptually mysterious and unclear why it works. Shout-out to people uncovering its mysteries (Dal Lago et al.[20], Borthelle et al. [21], Hirschowiz and Lafont [13]).

Howe's method in higher-order Abstract GSOS, briefly

Motivation: We need congruence of applicative similarity, not of its strong version.

Plan: Redo Howe's method using our abstract machinery, such that:

- We systematize it into a generic, language-independent method.
- Expose its core ideas, its language-specific part, and then simplify [2].

Key concept: Howe's closure $\hat{R}$: initial algebra (lfp) of an endofunctor on $\mathrm{Rel}_{\mu\Sigma}(\mathcal{C})$!

$$\text{For an applicative simulation } R \rightarrowtail \mu\Sigma \times \mu\Sigma, \quad \hat{R} = \mu S.\, R \vee \iota_\star[\overline{\Sigma}(S)]; R.$$

Results [22]: For $\hat{R}$ to be a bisimulation, just show that "weakened" rules are sound:

$$\frac{t \overset{s}{\Rightarrow} t'}{t\, s \Rightarrow t'} \;\checkmark \qquad \frac{t \Rightarrow t'}{t\, s \Rightarrow t'\, s} \;\checkmark \qquad \text{(Many thanks to dinaturality.)}$$

[2] Howe's method is complex, clunky, conceptually mysterious and unclear why it works. Shout-out to people uncovering its mysteries (Dal Lago et al.[20], Borthelle et al. [21], Hirschowiz and Lafont [13]).

Step-indexed Logical Relations

Time for some efficient reasoning in the Higher-order Abstract GSOS framework!

Question marks

Let's go over the typical setting of Logical Relations.

Concrete/Abstract

Question marks

Let's go over the typical setting of Logical Relations.

Concrete/Abstract

8. Operational Semantics

Question marks

Let's go over the typical setting of Logical Relations.

Concrete/Abstract

8. Operational Semantics

8. HO Abstract GSOS

Question marks

Let's go over the typical setting of Logical Relations.

Concrete/Abstract

Concrete	Abstract
8. Operational Semantics	8. HO Abstract GSOS
9. What is a Logical Relation?	

Question marks

Let's go over the typical setting of Logical Relations.

Concrete/Abstract

Concrete	Abstract
8. Operational Semantics	8. HO Abstract GSOS
9. What is a Logical Relation?	9. $R \leq (h \times \tilde{h})^{\star}[\overline{B}(R, R)]$

Question marks

Let's go over the typical setting of Logical Relations.

Concrete/Abstract

8. Operational Semantics
9. What is a Logical Relation?
10. Construct **that** Logical Relation, **the chosen one**

8. HO Abstract GSOS
9. $R \leq (h \times \tilde{h})^{\star}[\overline{B}(R, R)]$

Question marks

Let's go over the typical setting of Logical Relations.

Concrete/Abstract

8. Operational Semantics
9. What is a Logical Relation?
10. Construct **that** Logical Relation, **the chosen one**

8. HO Abstract GSOS
9. $R \leq (h \times \tilde{h})^{\star}[\overline{B}(R, R)]$
10. `Abstract Construction Missing`

Question marks

Let's go over the typical setting of Logical Relations.

Concrete/Abstract

8. Operational Semantics
9. What is a Logical Relation?
10. Construct **that** Logical Relation, **the chosen one**
11. Laborious compatibility lemmas

8. HO Abstract GSOS
9. $R \leq (h \times \tilde{h})^{\star}[\overline{B}(R, R)]$
10. `Abstract Construction Missing`

Question marks

Let's go over the typical setting of Logical Relations.

Concrete/Abstract

Concrete	Abstract
8. Operational Semantics	8. HO Abstract GSOS
9. What is a Logical Relation?	9. $R \leq (h \times \tilde{h})^{\star}[\overline{B}(R, R)]$
10. Construct **that** Logical Relation, **the chosen one**	10. `Abstract Construction Missing`
11. Laborious compatibility lemmas	11. ???

Question marks

Let's go over the typical setting of Logical Relations.

Concrete/Abstract

Concrete	Abstract
8. Operational Semantics	8. HO Abstract GSOS
9. What is a Logical Relation?	9. $R \leq (h \times \tilde{h})^{\star}[\overline{B}(R, R)]$
10. Construct **that** Logical Relation, **the chosen one**	10. `Abstract Construction Missing`
11. Laborious compatibility lemmas	11. ???
12. Reflexivity	

Question marks

Let's go over the typical setting of Logical Relations.

Concrete/Abstract

Concrete	Abstract
8. Operational Semantics	8. HO Abstract GSOS
9. What is a Logical Relation?	9. $R \leq (h \times \tilde{h})^{\star}[\overline{B}(R, R)]$
10. Construct **that** Logical Relation, **the chosen one**	10. `Abstract Construction Missing`
11. Laborious compatibility lemmas	11. ???
12. Reflexivity	12. General induction principle

Constructing step-indexed logical relation, Coalgebraically

In standard settings, (step-indexed) logical relations are defined empirically, on a per-case basis. Our approach systematizes the method. Let's see how:

Constructing step-indexed logical relation, Coalgebraically

In standard settings, (step-indexed) logical relations are defined empirically, on a per-case basis. Our approach systematizes the method. Let's see how:

Step-indexed Henceforth Relation Transformer

Let $B\colon \mathcal{C}^{\mathsf{op}} \times \mathcal{C} \to \mathcal{C}$ with a relation lifting $\overline{B}$, and let $c, \tilde{c}\colon X \to B(X,X)$ be coalgebras. For every $R \rightarrowtail X \times X$ we define the step-indexed logical relation $(\,\square^{\overline{B},c,\tilde{c},\alpha} R \rightarrowtail X \times X\,)_{\alpha}$ by transfinite induction (writing $\square^{\alpha}$ for simplicity):

$$\begin{aligned} \square^{0} R &= R, \\ \square^{\alpha+1} R &= \square^{\alpha} R \wedge (c \times \tilde{c})^{\star}[\overline{B}(\square^{\alpha} R, \square^{\alpha} R)], \\ \square^{\alpha} R &= \bigwedge_{\beta<\alpha} \square^{\beta} R \qquad \text{for limit ordinals } \alpha. \end{aligned}$$

Constructing step-indexed logical relation, Coalgebraically

In standard settings, (step-indexed) logical relations are defined empirically, on a per-case basis. Our approach systematizes the method. Let's see how:

Step-indexed Henceforth Relation Transformer

Let $B\colon \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathcal{C}$ with a relation lifting $\overline{B}$, and let $c, \tilde{c}\colon X \to B(X,X)$ be coalgebras. For every $R \rightarrowtail X \times X$ we define the step-indexed logical relation $(\,\square^{\overline{B},c,\tilde{c},\alpha} R \rightarrowtail X \times X\,)_\alpha$ by transfinite induction (writing $\square^\alpha$ for simplicity):

$$\begin{aligned} \square^0 R &= R, \\ \square^{\alpha+1} R &= \square^\alpha R \wedge (c \times \tilde{c})^\star[\overline{B}(\square^\alpha R, \square^\alpha R)], \\ \square^\alpha R &= \bigwedge_{\beta<\alpha} \square^\beta R \qquad \text{for limit ordinals } \alpha. \end{aligned}$$

Under mild conditions, there exists ν with $\square^{\nu+1} R = \square^\nu R$, which makes $\square^\nu R$ logical. For **the** logical relation, the "chosen one", plug $R = \top = X \times X$.

Constructing step-indexed logical relation, Coalgebraically

$$
\begin{aligned}
\mathcal{L}^0_\tau(\Gamma) &= \top_\tau(\Gamma) = \{(t,s) \mid \Gamma \vdash t, s \colon \tau\} \\
\mathcal{L}^{\alpha+1}_\tau &= \mathcal{L}^\alpha_\tau \cap \mathcal{S}_\tau(\mathcal{L}^\alpha, \mathcal{L}^\alpha) \cap \mathcal{E}_\tau(\mathcal{L}^\alpha) \cap \mathcal{V}_\tau(\mathcal{L}^\alpha, \mathcal{L}^\alpha) \\
\mathcal{L}^\alpha_\tau(\Gamma) &= \bigcap_{\beta<\alpha} \mathcal{L}^\alpha_\tau(\Gamma) \quad \text{for limit ordinals } \alpha. \\
\mathcal{S}_\tau(\Gamma)(Q,R) &= \{(t,s) \mid \text{for all } \Delta \text{ and } Q_{\Gamma(x)}(\Delta)(u_x, v_x)\ (x \in |\Gamma|), \\
&\qquad \text{one has } R_\tau(\Delta)(t[\vec{u}], s[\vec{v}])\}, \\
\mathcal{E}_\tau(\Gamma)(R) &= \{(t,s) \mid \text{if } t \to t' \text{ then } \exists s'.\, s \Rightarrow s' \wedge R_\tau(\Gamma)(t', s')\}, \\
\mathcal{V}_{\tau_1 \boxtimes \tau_2}(\Gamma)(Q,R) &= \{(t,s) \mid \text{if } t = \mathsf{pair}_{\tau_1,\tau_2}(t_1,t_2) \text{ then } \exists s_1, s_2.\, s \Rightarrow \mathsf{pair}_{\tau_1,\tau_2}(s_1,s_2) \wedge \\
&\qquad R_{\tau_1}(\Gamma)(t_1,s_1) \wedge R_{\tau_2}(\Gamma)(t_2,s_2)\}, \\
\mathcal{V}_{\mu\alpha.\tau}(\Gamma)(Q,R) &= \{(t,s) \mid \text{if } t = \mathsf{fold}_\tau(t') \text{ then } \exists s'.\, s \Rightarrow \mathsf{fold}_\tau(s') \wedge R_{\tau[\mu\alpha.\tau/\alpha]}(\Gamma)(t',s')\}, \\
\mathcal{V}_{\tau_1 \multimap \tau_2}(\Gamma)(Q,R) &= \{(t,s) \mid \text{for all } Q_{\tau_1}(\Gamma)(e,e'), \\
&\qquad \text{if } t = \lambda x.t' \text{ then } \exists s'.\, s \Rightarrow \lambda x.s' \ \wedge\ R_{\tau_2}(\Gamma)(t'[e/x], s'[e'/x])\}.
\end{aligned}
$$

Some results

Data: Higher-Order GSOS law of Σ over B in a suitable category $\mathcal{C}$, liftings, weakening of the operational model (the coalgebra on terms $\mu\Sigma$) and mild conditions on $\mathcal{C}$.

Main theorem (informal)

Let $R \rightarrowtail \mu\Sigma \times \mu\Sigma$ be a congruence. Assuming a lax-bialgebra condition. If R is a congruence, then for all α, $\Box^\alpha R$ is a congruence.

$$\frac{t \overset{s}{\Rightarrow} t'}{t\, s \Rightarrow t'} \quad \checkmark \qquad\qquad \frac{t \Rightarrow t'}{t\, s \Rightarrow t'\, s} \quad \checkmark$$

Some results

Data: Higher-Order GSOS law of Σ over B in a suitable category $\mathcal{C}$, liftings, weakening of the operational model (the coalgebra on terms $\mu\Sigma$) and mild conditions on $\mathcal{C}$.

Main theorem (informal)

Let $R \rightarrowtail \mu\Sigma \times \mu\Sigma$ be a congruence. Assuming a lax-bialgebra condition. If R is a congruence, then for all α, $\Box^\alpha R$ is a congruence.

$$\frac{t \overset{s}{\Rightarrow} t'}{t\, s \Rightarrow t'} \quad \checkmark \qquad \frac{t \Rightarrow t'}{t\, s \Rightarrow t'\, s} \quad \checkmark$$

Corollary

1. *For all α, $\Box^\alpha \top$ is a congruence.*
2. *$\Box^\nu \top$ is a congruence (and hence reflexive) and, for "reasonable" definitions of contextual equivalence, sound w.r.t. contextual equivalence.*

The point of all this

Accept for a single slide that every higher-order operational semantics is a higher-order GSOS law. You are presented with such semantics and looking for a sensible logical relation, sound for contextual equivalence. What we're saying is that the actual work that needs to be done, the language-dependent part of the problem, is the following:

The point of all this

Accept for a single slide that every higher-order operational semantics is a higher-order GSOS law. You are presented with such semantics and looking for a sensible logical relation, sound for contextual equivalence. What we're saying is that the actual work that needs to be done, the language-dependent part of the problem, is the following:

1. Decide what kinds of relational reasoning you're looking for (define the lifting $\overline{B}$).

The point of all this

Accept for a single slide that every higher-order operational semantics is a higher-order GSOS law. You are presented with such semantics and looking for a sensible logical relation, sound for contextual equivalence. What we're saying is that the actual work that needs to be done, the language-dependent part of the problem, is the following:

1. Decide what kinds of relational reasoning you're looking for (define the lifting $\overline{B}$).
2. Check that your notion of "weakening" is sensible w.r.t. the operational semantics.

The point of all this

Accept for a single slide that every higher-order operational semantics is a higher-order GSOS law. You are presented with such semantics and looking for a sensible logical relation, sound for contextual equivalence. What we're saying is that the actual work that needs to be done, the language-dependent part of the problem, is the following:

1. Decide what kinds of relational reasoning you're looking for (define the lifting $\overline{B}$).
2. Check that your notion of "weakening" is sensible w.r.t. the operational semantics.

The intuition is that the standard compatibility lemmas contain lots of boilerplate, contrived proof code that should be "automatic" under reasonable circumstances.

The point of all this

It's not just that higher-order abstract GSOS is cool and efficient. By systematizing (step-indexed) logical relations, we show that, assuming the operational semantics are minimally sane, the evident logical relation should be reflexive and sound w.r.t. contextual equivalence.

Logical Predicates

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

$$
\begin{aligned}
\mathrm{SN}_{\mathsf{unit}}(t) &= \Downarrow_{\mathsf{unit}}(t) \\
\mathrm{SN}_{\tau_1 \multimap\!\!\!\!\rightarrow \tau_2}(t) &= \Downarrow_{\tau_1 \multimap\!\!\!\!\rightarrow \tau_2}(t) \land (\forall s \colon \tau_1.\, \mathrm{SN}_{\tau_1}(s) \implies \mathrm{SN}_{\tau_2}(t \cdot s))
\end{aligned}
$$

Strong Normalization

Definition (A standard logical predicate)

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

Definition (Open extension of 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 λ-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 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 $(\lambda x\colon \tau_1 . t') \cdot s$ with $\mathrm{SN}_{\tau_1}(s)$ 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

$$\begin{tikzcd}
\mathrm{SN} \arrow[r, "\text{Any predicate } P"] \arrow[d, "\text{Any semantics } \gamma"] & \Box P \arrow[d, dashed] \\
\mathrm{SN}^{\gamma} \arrow[r, dashed] & \Box^{\gamma} P
\end{tikzcd}$$

The goal of this talk

I will argue for two directions of abstraction, via

Higher-order Abstract GSOS

$$
\begin{tikzcd}
\mathrm{SN} \arrow[rr, "\text{Any predicate } P"] \arrow[d, "\text{Any semantics } \gamma"] & & \Box P \arrow[d, dashed] \\
\mathrm{SN}^{\gamma} \arrow[rr, dashed] & & \Box^{\gamma} P
\end{tikzcd}
$$

+Efficient reasoning!

Dissecting the logical predicate (1)

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

Dissecting the logical predicate (1)

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

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)

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

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

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

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)

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

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 $\Lambda_{\tau_1 \to\!\!\triangleright\, \tau_2}$

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

Induction up to $\boxdot$ on STLC

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* $\cancel{\square_{\mathrm{unit}} P}$ P_{unit},
2. *for all closed application terms* $t\,s$ *such that* $\square_{\tau_1 \rightarrow \tau_2} P(t)$ *and* $\square_{\tau_1} P(s)$, *we have* $\cancel{\square_{\tau_2} P(t\,s)}$ $P_{\tau_2}(t\,s)$, *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* $\square P$ *and given a substitution* $\vec{e}$ *that satisfies* $\square P$, $(\lambda x : \tau_1 .\, t)[\vec{e}/\vec{x}]$, *we have that* $(\lambda x : \tau_1 .\, t)[\vec{e}/\vec{x}]$ *is in* $\cancel{\square P}$, P.

Proof.

Instantiate [18, Th. 36] with $(\mathrm{Th36}.P)_\tau(\varnothing) = P_\tau$ and $(\mathrm{Th36}.P)_\tau(\Gamma \neq \varnothing) = \top$. $\square$

Let's try this out!

Proving strong normalization for STLC

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

Let's try this out!

Proving strong normalization for STLC

1. $\Downarrow_{\mathsf{unit}} (\mathsf{e})$;
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).

Proof.

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

Let's explore the other direction

$$\begin{tikzcd}
\mathrm{SN} \arrow[r, "\text{Any predicate } P"] \arrow[d, "\text{Any semantics } \gamma"] & \square P \arrow[d, dashed] \\
\mathrm{SN}^{\gamma} \arrow[r, dashed] & \square^{\gamma} P
\end{tikzcd}$$

+Efficient reasoning!

Objective Complete

Let's explore the other direction

$$
\begin{tikzcd}
\mathrm{SN} \arrow[r, "\text{Any predicate } P"] \arrow[d, "\text{Any semantics } \gamma"] & \Box P \arrow[d, dashed, no head] \\
\mathrm{SN}^{\gamma} \arrow[r, dashed] & \Box^{\gamma} P
\end{tikzcd}
$$

+Efficient reasoning!

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 $\square P$, 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 $\frac{t \to t'}{t \cdot s \to t' \cdot s}$

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

- (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.

- 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

- (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.

- 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.

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

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

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

$$\begin{tikzcd}
\mathrm{Pred}(\mathcal{C})^{\mathrm{op}} \times \mathrm{Pred}(\mathcal{C}) \arrow[r, "\overline{B}"] \arrow[d, "|-|^{\mathrm{op}} \times |-|"'] & \mathrm{Pred}(\mathcal{C}) \arrow[d, "|-|"] \\
\mathcal{C}^{\mathrm{op}} \times \mathcal{C} \arrow[r, "B"] & \mathcal{C}
\end{tikzcd}$$

Categorical machinery

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

$$\begin{tikzcd}
\mathrm{Pred}(\mathcal{C})^{\mathsf{op}} \times \mathrm{Pred}(\mathcal{C}) \arrow[r, "\overline{B}"] \arrow[d, "|-|^{\mathsf{op}} \times |-|"'] & \mathrm{Pred}(\mathcal{C}) \arrow[d, "|-|"] \\
\mathcal{C}^{\mathsf{op}} \times \mathcal{C} \arrow[r, "B"] & \mathcal{C}
\end{tikzcd}$$

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

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

Logical Predicate

A predicate $P \rightarrowtail \mu\Sigma$ is logical (for γ) 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 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:

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 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 $t \to t'$, 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 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 $t \to t'$, 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 $\Box$

The definition of logicality and $\Box$ 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 λ-**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!

Future Work

References

D. Turi, G. D. Plotkin, "Towards a mathematical operational semantics", in 12th Annual IEEE Symposium on Logic in Computer Science (LICS 1997), 1997, pp. 280–291. DOI: 10.1109/LICS.1997.614955.

M. P. Fiore, D. Turi, "Semantics of name and value passing", in 16th Annual IEEE Symposium on Logic in Computer Science (LICS 2001), IEEE Computer Society, 2001, pp. 93–104. DOI: 10.1109/LICS.2001.932486.

F. Bartels, "On generalised coinduction and probabilistic specification formats: Distributive laws in coalgebraic modelling", English, PhD thesis, Vrije Universiteit Amsterdam, 2004.

M. P. Fiore, S. Staton, "A congruence rule format for name-passing process calculi from mathematical structural operational semantics", in 21st Annual IEEE Symposium on Logic in Computer Science, LICS'06, IEEE Computer Society, 2006, pp. 49–58. DOI: 10.1109/LICS.2006.7. [Online]. Available: https://doi.org/10.1109/LICS.2006.7.

M. Miculan, M. Peressotti, "Structural operational semantics for non-deterministic processes with quantitative aspects", Theor. Comput. Sci., vol. 655, pp. 135–154, 2016. DOI: 10.1016/j.tcs.2016.01.012. [Online]. Available: https://doi.org/10.1016/j.tcs.2016.01.012.

B. Klin, V. Sassone, "Structural operational semantics for stochastic process calculi", in 11th International Conference Foundations of Software Science and Computational Structures, FOSSACS'08, R. M. Amadio, Ed., ser. LNCS, vol. 4962, Springer, 2008, pp. 428–442. DOI: 10.1007/978-3-540-78499-9_30. [Online]. Available: https://doi.org/10.1007/978-3-540-78499-9_30.

F. Abou-Saleh, D. Pattinson, "Towards effects in mathematical operational semantics", in Mathematical Foundations of Programming Semantics, MFPS 2011, M. W. Mislove, J. Ouaknine, Eds., ser. ENTCS, vol. 276, Elsevier, 2011, pp. 81–104. DOI: 10.1016/j.entcs.2011.09.016.

——, "Comodels and effects in mathematical operational semantics", in Foundations of Software Science and Computation Structures - 16th International Conference, FOSSACS 2013, Held as Part of the European Joint Con F. Pfenning, Ed., ser. Lecture Notes in Computer Science, vol. 7794, Springer, 2013, pp. 129–144. DOI: 10.1007/978-3-642-37075-5_9. [Online]. Available: https://doi.org/10.1007/978-3-642-37075-5_9.

S. Goncharov, S. Milius, L. Schröder, S. Tsampas, H. Urbat, "Stateful structural operational semantics", in 7th International Conference on Formal Structures for Computation and Deduction, FSCD'22, A. P. Felty, Ed., ser. LIPIcs, vol. 228, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 30:1–30:19. DOI: 10.4230/LIPIcs.FSCD.2022.30. [Online]. Available: https://doi.org/10.4230/LIPIcs.FSCD.2022.30.

H. Watanabe, "Well-behaved translations between structural operational semantics", Electr. Notes Theor. Comput. Sci., vol. 65, no. 1, pp. 337–357, 2002. DOI: 10.1016/S1571-0661(04)80372-4. [Online]. Available: https://doi.org/10.1016/S1571-0661(04)80372-4.

B. Klin, B. Nachyla, "Presenting morphisms of distributive laws", in 6th Conference on Algebra and Coalgebra in Computer Science, CALCO'15, ser. LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015, pp. 190–204. DOI: 10.4230/LIPIcs.CALCO.2015.190. [Online]. Available: https://doi.org/10.4230/LIPIcs.CALCO.2015.190.

S. Tsampas, A. Nuyts, D. Devriese, F. Piessens, "A categorical approach to secure compilation", in 15th IFIP WG 1.3 International Workshop on Coalgebraic Methods in Computer Science, CMCS'20, D. Petrisan, J. Rot, Eds., ser. LNCS, vol. 12094, Springer, 2020, pp. 155–179. DOI: 10.1007/978-3-030-57201-3_9.

T. Hirschowitz, A. Lafont, "A categorical framework for congruence of applicative bisimilarity in higher-order languages", Log. Methods Comput. Sci., vol. 18, no. 3, 2022. DOI: 10.46298/lmcs-18(3:37)2022. [Online]. Available: https://doi.org/10.46298/lmcs-18(3:37)2022.

H. B. Curry, "Grundlagen der kombinatorischen Logik", Am. J. Math., vol. 52, no. 3, pp. 509–536, 1930, ISSN: 00029327, 10806377. [Online]. Available: http://www.jstor.org/stable/2370619 (visited on 05/18/2022).

M. P. Fiore, G. D. Plotkin, D. Turi, "Abstract syntax and variable binding", in 14th Annual IEEE Symposium on Logic in Computer Science (LICS 1999), IEEE Computer Society, 1999, pp. 193–202. DOI: 10.1109/LICS.1999.782615.

Bibliography iv

S. Goncharov, S. Milius, L. Schröder, S. Tsampas, H. Urbat, "Towards a higher-order mathematical operational semantics", in 50th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2023), ser. Proc. ACM Program. Lang. Vol. 7, ACM, 2023. DOI: 10.1145/3571215.

A. Kurz, J. Velebil, "Relation lifting, a survey", Journal of Logical and Algebraic Methods in Programming, Relational and Algebraic Methods in Computer Science, vol. 85, no. 4, pp. 475–499, 2016, ISSN: 2352-2208. DOI: 10.1016/j.jlamp.2015.08.002.

S. Goncharov, A. Santamaria, L. Schröder, S. Tsampas, H. Urbat, "Logical predicates in higher-order mathematical operational semantics",, N. Kobayashi, J. Worrell, Eds., 2024.

S. Goncharov, S. Milius, S. Tsampas, H. Urbat, "Bialgebraic reasoning on higher-order program equivalence", in 39th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2024), Preprint: https://arxiv.org/abs/2402.00625, IEEE Computer Society Press, 2024. DOI: 10.1145/3661814.3662099.

U. Dal Lago, F. Gavazzo, P. B. Levy, "Effectful applicative bisimilarity: Monads, relators, and Howe's method", in 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2017), IEEE Computer Society, 2017, pp. 1–12. DOI: 10.1109/LICS.2017.8005117.

P. Borthelle, T. Hirschowitz, A. Lafont, "A cellular Howe theorem", in 35th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS'20, H. Hermanns, L. Zhang, N. Kobayashi, D. Miller, Eds., ACM, 2020, pp. 273–286. DOI: 10.1145/3373718.3394738. [Online]. Available: https://doi.org/10.1145/3373718.3394738.

Bibliography v

H. Urbat, S. Tsampas, S. Goncharov, S. Milius, L. Schröder, "Weak similarity in higher-order mathematical operational semantics", in 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2023), IEEE Computer Society Press, 2023. DOI: 10.1109/LICS56636.2023.10175706.