# Towards Light-Weight Operational Reasoning for Languages with Binders, Categorically

**Sergey Goncharov**

University of Birmingham

HOPE 2026, August 24, Indianapolis

# Operational Reasoning about Higher-Order Programs

Contextual equivalence: $p \simeq_{ctx} q$ = "$C[p]$ and $C[q]$ behave the same, for every context $C$"

**Challenge:** Prove $p \simeq_{ctx} q$ for given $p$ and $q$ (problematic because of $\forall C$)

**Established methods:**

- Applicative bisimilarity[†] + Howe's method[‡]
- Logical relations, step-indexing, environmental bisimulations, etc

Each method is re-developed, and re-proved sound, per language and per language feature

**Other challenges:** strong normalisation, type safety, free theorems, behavioural metrics, ...

[†] Abramsky, "The lazy $\lambda$-calculus", 1990.

[‡] Howe, "Proving Congruence of Bisimulation in Functional Programming Languages", 1996.

# Categorical Operational Semantics Project

- **Main idea:** given operational specification (set of rules), devise methods/properties of the language, using category theory as leverage


  - **Methods:** abstract logical relations, abstract Howe's method, ...
  - **Properties:** compositionality, safety, adequacy, equivalence of programs, semantic styles
- **Side-effect:** categorical methods $\rightsquigarrow$ functional implementation (Haskell, Agda, Rocq)

# Higher-Order Mathematical Operational Semantics

- Goncharov, Milius, Schröder, Tsampas, and Urbat, "Towards a Higher-Order Mathematical Operational Semantics", POPL 2023
- Urbat, Tsampas, Goncharov, Milius, and Schröder, "Weak Similarity in Higher-Order Mathematical Operational Semantics", LICS 2023
- Goncharov, Santamaria, Schröder, Tsampas, and Urbat, "Logical Predicates in Higher-Order Mathematical Operational Semantics", FoSSaCS 2024
- Goncharov, Milius, Tsampas, and Urbat, "Bialgebraic Reasoning on Higher-Order Program Equivalence", LICS 2024
- Goncharov, Tsampas, and Urbat, "Abstract Operational Methods for Call-by-Push-Value", POPL 2025
- Goncharov, Milius, Schröder, Tsampas, and Urbat, "Bialgebraic Reasoning on Stateful Languages", ICFP 2025
- Goncharov, Partow, and Tsampas, "Big Steps in Higher-Order Mathematical Operational Semantics", ICFP 2025
- Urbat, "Higher-Order Behavioural Conformances via Fibrations", POPL 2026

.. and rolling

- **Goncharov, Peressotti, Tsampas, Urbat, and Volpe, "Towards a Higher-Order Bialgebraic Denotational Semantics", ICFP 2026**

# Binders, Categorically

- Categorical modelling of languages with variable binders is usually taken to require category that internalises names and substitution
  - Categories of presheaves $[\mathbb{F}, \mathbf{Set}]$[†], or nominal sets[‡]

☺ Mathematically natural, well-motivated for study of syntax

☹ Technically demanding, hard to track precisely in implementations

? Is this machinery really necessary, if goal is treatment of operational semantics rather than of syntax as such?

[†] Fiore, Plotkin, and Turi, "Abstract Syntax and Variable Binding", 1999.
[‡] Pitts, *Nominal Sets: Names and Symmetry in Computer Science*, 2013.

# Binders, Categorically

- Categorical modelling of languages with variable binders is usually taken to require category that internalises names and substitution
  - Categories of presheaves $[\mathbb{F}, \mathbf{Set}]$[†], or nominal sets[‡]

☺ Mathematically natural, well-motivated for study of syntax

☹ Technically demanding, hard to track precisely in implementations

? Is this machinery really necessary, if goal is treatment of operational semantics rather than of syntax as such?

**Alternative View:** Substitution is something that *happens* during execution — so it can be treated as an operational action, rather than built into the model

[†]Fiore, Plotkin, and Turi, "Abstract Syntax and Variable Binding", 1999.
[‡]Pitts, *Nominal Sets: Names and Symmetry in Computer Science*, 2013.

# Plan

- **Part I:** Higher-order abstract GSOS, categorical operational model and generic congruence theorem
  - Extended combinatory logic as running example
- **Part II:** From languages with binders to languages without binders
  - Combinatorial $\lambda$-calculus: $\lambda$-terms as combinators, substitution suspended in the term
  - Combinatorial version behaves equivalently to Abramsky's lazy $\lambda$-calculus under strong (!) applicative bisimilarity
  - Congruence for lazy $\lambda$-calculus follows categorically over **Set** — no presheaves or the like
- **Prospects:** Work in progress; ideas towards general result

# Higher-Order Abstract GSOS

# First-Order Abstract GSOS

Turi and Plotkin's abstraction of GSOS rule format[†]:

- Signature endofunctor $\Sigma$
- Behaviour endofunctor $B$
- GSOS law — natural transformation $\rho_X \colon \Sigma(X \times BX) \to B(\Sigma^\star X)$

**Example (process algebra):**

- $\Sigma = \{\,|\colon 2, \emptyset\colon 0\} \cup \{a.(-)\colon 1 \mid a \in A\}$
- $BX = \mathcal{P}(A \times X)$
- $\rho$ encodes rules like:

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

[†]Turi and Plotkin, "Towards a Mathematical Operational Semantics", 1997.

# First-Order Abstract GSOS

Turi and Plotkin's abstraction of GSOS rule format[†]:

- Signature endofunctor $\Sigma$
- Behaviour endofunctor $B$
- GSOS law — natural transformation $\rho_X \colon \Sigma(X \times BX) \to B(\Sigma^\star X)$

**Example (process algebra):**

- $\Sigma = \{\ |\colon 2, \emptyset\colon 0\} \cup \{a.(-)\colon 1 \mid a \in A\}$
- $BX = \mathcal{P}(A \times X)$
- $\rho$ encodes rules like:

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

operation from $\Sigma$

behaviour from $B$

[†]Turi and Plotkin, "Towards a Mathematical Operational Semantics", 1997.

# First-Order Abstract GSOS

Turi and Plotkin's abstraction of GSOS rule format[†]:

- Signature endofunctor $\Sigma$
- Behaviour endofunctor $B$
- GSOS law — natural transformation $\rho_X \colon \Sigma(X \times BX) \to B(\Sigma^\star X)$

**Example (process algebra):**

- $\Sigma = \{\,|\colon 2, \emptyset\colon 0\} \cup \{a.(-)\colon 1 \mid a \in A\}$
- $BX = \mathcal{P}(A \times X)$
- $\rho$ encodes rules like:

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

[†]Turi and Plotkin, "Towards a Mathematical Operational Semantics", 1997.

# Bialgebras and Operational Model

- $\rho$-bialgebra consists of algebra $a\colon \Sigma A \to A$, coalgebra $c\colon A \to BA$ and coherence condition: bialgebra law (omitted)
- Initial algebra $(\mu\Sigma, \iota)$ of program terms extends to initial bialgebra, with $\gamma\colon \mu\Sigma \to B\mu\Sigma$ — operational model
  - This induces operational bisimilarity relation $\sim \subseteq \mu\Sigma \times \mu\Sigma$
- Final coalgebra $\nu B$ extends to final bialgebra — denotational model
- Unique bialgebra morphism

$$[\![-]\!]_\rho\colon \mu\Sigma \to \nu B$$

Hence $p \sim q$ iff $[\![p]\!]_\rho = [\![q]\!]_\rho$

- $\sim$ is a congruence by abstract nonsense: $p \sim q \Rightarrow C[p] \sim C[q]$ for any context

# Higher-Order GSOS

Higher-order GSOS law in category $\mathbf{C}$ consists of

- Signature functor $\Sigma\colon \mathbf{C} \to \mathbf{C}$
- Behaviour bifunctor $B\colon \mathbf{C}^{\mathrm{op}} \times \mathbf{C} \to \mathbf{C}$
- Family

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

  natural in $Y$, dinatural in $X$

First argument of $B$ is contravariant: program can be observed by feeding it other programs

# Extended Combinatory Logic xCL

- Combinators $K$, $S$, $I$
- .. plus "multiary combinators" $K'$, $S'$, $S''$ for partial applications
- $\mathbf{C} = \mathbf{Set}$, $\Sigma$ = signature functor of $\{S, S', S'', K, K', I, \mathsf{app}\}$, $B(X,Y) = Y + Y^X$:

$$S \xrightarrow{t} S'(t) \qquad S'(t_1) \xrightarrow{t_2} S''(t_1,t_2) \qquad S''(t_1,t_2) \xrightarrow{t} t_1\, t\, (t_2\, t)$$

$$K \xrightarrow{t} K'(t) \qquad K'(t_1) \xrightarrow{t_2} t_1 \qquad I \xrightarrow{t} t \qquad \frac{s \to s'}{s\, t \to s'\, t} \qquad \frac{s \xrightarrow{t} s'}{s\, t \to s'}$$

**Example:** $S\, p\, q\, r \to S'(p)\, q\, r \to S''(p,q)\, r \to (p\, r)(q\, r)$

# Rules as GSOS Law

Rules

$$\frac{s \to s'}{s\ t \to s'\ t} \qquad \frac{s \xrightarrow{t} s'}{s\ t \to s'}$$

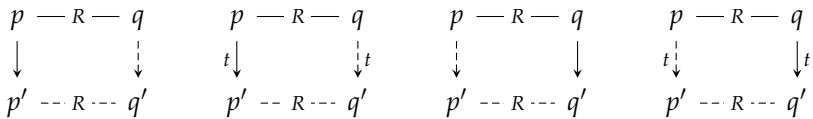
correspond to

$$\rho((s, s')(t, \_)) = s'\ t$$
$$\rho((s, f)(t, \_)) = f(t)$$

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

# Higher-Order Bialgebras and Congruence

- Higher-order $\rho$-bialgebra: algebra $a\colon \Sigma A \to A$, coalgebra $c\colon A \to B(A, A)$ and (now) higher-order bialgebra law (omitted)
- Initial algebra $(\mu\Sigma, \iota)$ extends to the initial bialgebra, with $\gamma\colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$ — operational model
- For xCL, induced $\sim$ is strong applicative bisimilarity: greatest $R \subseteq \mu\Sigma \times \mu\Sigma$ closed under




! Final bialgebra need not exist: $\sim$ is kernel of $\mathrm{coit}(\gamma)\colon \mu\Sigma \to \nu\gamma.\, B(\mu\Sigma, \gamma)$, no longer of map into denotational model

- See my ICFP 2026 talk on this

# Abstract Congruence for Higher-Order GSOS

**Theorem**[†]**:** given that

1. $\mathbf{C}$ is regular
2. $\Sigma$ preserves reflexive coequalizers ($\approx \Sigma$ is finitary)
3. $B$ preserves monomorphisms in both arguments

strong bisimilarity $\sim$ on $\mu\Sigma$ is a congruence

$\rightsquigarrow$ for xCL: strong applicative bisimilarity is a congruence, for free

**!** This is relevant because congruence of applicative bisimilarity (albeit weak) is established tool for proving contextual equivalences

[†]Goncharov, Milius, Schröder, Tsampas, and Urbat, "Towards a Higher-Order Mathematical Operational Semantics", 2023.

# Binders and Combinators

# Call-by-Name (Lazy) $\lambda$-calculus

Operational (small-step, call-by-name) semantics rules

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

We can reframe the first rule ($\beta$-reduction) as:

$$\frac{}{\lambda x.\, p \xrightarrow{q} p[q/x]}$$

to fit GSOS format. But this is not enough!

# Lazy $\lambda$-Calculus as HO-GSOS

**Idea:** Refactor $\beta$-reduction as

$$\frac{p \xrightarrow{[q/x]} p'}{\lambda x.\, p \xrightarrow{q} p'}$$

involving "substitution transitions" $\xrightarrow{[q/x]}$

- $\mathbf{C} = \mathbf{Set}^{\mathbb{F}}$, where $\mathbb{F}$ is category of finite cardinals

$$\Sigma\colon \mathbf{C} \to \mathbf{C}, \qquad \Sigma X = V + \delta X + X \times X,$$

$$B\colon \mathbf{C}^{\mathrm{op}} \times \mathbf{C} \to \mathbf{C}, \qquad B(X,Y) = \langle\!\langle X, Y \rangle\!\rangle \times (Y + Y^X + 1)$$

($V$ is presheaf of variables $V(n) = n$, $(\delta X)(n) = X(n+1)$, $\langle\!\langle X, Y \rangle\!\rangle(n) = \mathbf{Set}^{\mathbb{F}}(X^n, Y)$)

- $\mu\Sigma$ is presheaf $\Lambda \in \mathbf{Set}^{\mathbb{F}}$ of $\lambda$-terms over $n$ free variables

How much of this is needed for operational semantics?

# From xCL to $\lambda$-Calculus

Covering binders with HO-GSOS is possible, but adds friction, which scales henceforth

Treat every $\lambda$-term as (multiary) combinator

For each $p \in \Lambda(n{+}1)$ ($\lambda$-terms with free variables among $1, \ldots, n{+}1$), introduce $n$-ary operation $(\!|p|\!)$:

$$\Sigma \;=\; \{\mathsf{app}\colon 2\} \;\cup\; \{(\!|p|\!)\colon n \mid n \in \mathbb{N},\; p \in \Lambda(n{+}1)\}$$

Reading: $(\!|p|\!)(t_1, \ldots, t_n)$ represents substituted term $(\lambda n{+}1.\, p)[t_1/1, \ldots, t_n/n]$

This is closure in sense of SECD machine: code $p$ together with environment $[t_1/1, \ldots, t_n/n]$. Substitution is suspended until argument arrives

# Rules of Combinatorial $\lambda$-Calculus

$$\frac{}{(\!|p|\!)(t_1,\ldots,t_n) \xrightarrow{t} (\!|p|\!)(t_1,\ldots,t_n,t)} \quad (p \in \Lambda(n{+}1)) \qquad \frac{s \to s'}{s\,t \to s'\,t} \qquad \frac{s \xrightarrow{t} s'}{s\,t \to s'}$$

where $\quad (\!|i|\!)(\vec{t}) = t_i \qquad (\!|pq|\!)(\vec{t}) = (\!|p|\!)(\vec{t})\,(\!|q|\!)(\vec{t}) \qquad (\!|\lambda x.\, p|\!)(\vec{t}) = (\!|p[n{+}1/x]|\!)(\vec{t})$

- Every closed $\lambda$-term $p \in \Lambda(0)$ embeds as constant $(\!|p|\!) \in \mu\Sigma$
- Conversely, $\tau\colon \mu\Sigma \to \Lambda(0)$ folds combinatorial term back down:

$$\tau(s\,t) = \tau(s)\,\tau(t) \qquad \tau((\!|p|\!)(t_1,\ldots,t_n)) = \lambda n{+}1.\, p[\tau(t_1)/1,\ldots,\tau(t_n)/n]$$

- $\tau$ preserves and reflects one-step reductions, and $\tau((\!|p|\!)) = p$

# Main Result

**Theorem:** let $\sim$ be canonical bisimilarity on $\mu\Sigma$ (induced by rules above), and $\sim_n^\Lambda$ strong applicative bisimilarity as before. Then

1. for all $s,t \in \mu\Sigma$: $\quad s \sim t \quad$ iff $\quad \tau(s) \sim_0^\Lambda \tau(t)$
2. for all $p,q \in \Lambda(0)$: $\quad p \sim_0^\Lambda q \quad$ iff $\quad (\!|p|\!) \sim (\!|q|\!)$

**!** This agreement is up to strong bisimilarity — reductions agree step-wise.

This is much sharper than mere computational equivalence of $\lambda$-calculus and combinatory logic

**Example:** $I\,I \to I$, while $I$ is value, hence $I \not\sim_0^\Lambda I\,I$; likewise $(\!|I|\!) = (\!|1|\!) \not\sim (\!|1|\!)\,(\!|1|\!) = (\!|I\,I|\!)$

# Main Result

**Theorem:** let $\sim$ be canonical bisimilarity on $\mu\Sigma$ (induced by rules above), and $\sim_n^\Lambda$ strong applicative bisimilarity as before. Then

1. for all $s,t \in \mu\Sigma$: $\quad s \sim t \quad$ iff $\quad \tau(s) \sim_0^\Lambda \tau(t)$
2. for all $p,q \in \Lambda(0)$: $\quad p \sim_0^\Lambda q \quad$ iff $\quad (\!|p|\!) \sim (\!|q|\!)$

**!** This agreement is up to strong bisimilarity — reductions agree step-wise.

This is much sharper than mere computational equivalence of $\lambda$-calculus and combinatory logic

**Example:** $I\,I \to I$, while $I$ is value, hence $I \not\sim_0^\Lambda I\,I$; likewise $(\!|I|\!) = (\!|1|\!) \not\sim (\!|1|\!)\,(\!|1|\!) = (\!|I\,I|\!)$

**Corollary:** Congruence transfers along $\tau/(\!|\cdot|\!)$ to lambda-calculus

# Towards a General Result

- Theorem above is one instance; goal is general result it instantiates for broader class of languages with binders
- **Route:** refine higher-order GSOS to separate
  - purely algebraic part, governed by standard higher-order GSOS law, from
  - part describing substitution behaviour

# Open Connections

- **Closures / environments:** is there general recipe — compile *any* language with binders into HO-GSOS-shaped "closure calculus", for other evaluation strategies too (call-by-value, call-by-need, ...), and relate resulting bisimilarities?
- **Explicit substitutions:** $(\!|p|\!)(t_1, \ldots, t_n)$ resembles $\lambda$-calculi with explicit substitutions[†]
  - can such calculi themselves be captured by higher-order GSOS law in **Set**?
- **Presheaves:** closure-based evaluators are defunctionalised domain models of $\lambda$-calculus — what is precise bridge to presheaf-based account of binders[‡]?
- **Effects:** how does picture change once language has computational effects (nondeterminism, store, ...) alongside binders?

[†] Abadi, Cardelli, Curien, and Lévy, "Explicit Substitutions", 1991.

[‡] Fiore, "Second-Order and Dependently-Sorted Abstract Syntax", 2008; Fiore, Plotkin, and Turi, "Abstract Syntax and Variable Binding", 1999.

# Summary

- Binding and substitution need not live in ambient categorical model — they can be pushed into operational rules themselves
- For lazy $\lambda$-calculus, this recovers strong applicative bisimilarity exactly, and its congruence property comes for free from higher-order GSOS in **Set**
- Several open threads: explicit substitutions, presheaves, other evaluation strategies, effects

Thank you!