From Small-Step to Big-Step, Abstractly

Sergey Goncharov (University of Birmingham)

(joint work with Pouya Partow & Stelios Tsampas)

UCL, PPLV, April 3, 2025, London

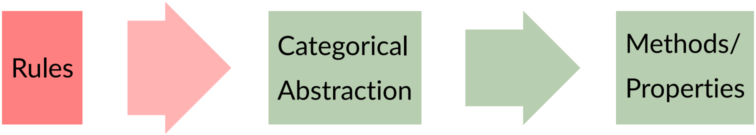
HO Mathematical Operational Semantics Project

- Goncharov, Milius, Schröder, Tsampas, and Urbat, "Towards a Higher-Order Mathematical Operational Semantics", 2023, POPL 2023
- Urbat, Tsampas, Goncharov, Milius, and Schröder, "Weak Similarity in Higher-Order Mathematical Operational Semantics", 2023, LICS 2023
- Goncharov, Santamaria, Schröder, Tsampas, and Urbat, "Logical Predicates in Higher-Order Mathematical Operational Semantics", 2024, FoSSaCS 2024
- Goncharov, Milius, Tsampas, and Urbat, "Bialgebraic Reasoning on Higher-Order Program Equivalence", 2024, LICS 2024
- Goncharov, Tsampas, and Urbat, "Abstract Operational Methods for Call-by-Push-Value", 2025, POPL 2025

.. and rolling

Project in Nutshell

- **Motto:** Make operational semantics more mathematical
- **Main Idea:** Given operational specification (set of O/S rules), devise methods/properties of the language
- **Main Tool:** Category theory



  - **Methods:** Abstract logical relations, Abstract Howe's method
  - **Properties:** Compositionality, safety, adequacy
- **Side-effect:** categorical methods $\rightsquigarrow$ functional implementation (Haskell, Agda, Coq)

Our Hobbyhorse: (Extended) Combinatory Logic

- I (=$\lambda p.\, p$) $\quad K$ (=$\lambda p.\lambda q.\, p$) $\quad S$ (=$\lambda p.\lambda q.\lambda r.\, (p \cdot r) \cdot (q \cdot r)$)
- plus S', S'' and K' for partially reduced terms

Small-step semantics (cf. lazy λ-calculus[†]):

$$K \xrightarrow{p} K'(p) \qquad K'(p) \xrightarrow{q} p \qquad S''(p,q) \xrightarrow{r} (p \cdot r) \cdot (q \cdot r) \qquad \ldots$$

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

Example:

$$\cfrac{\cfrac{K \xrightarrow{S} K'(S)}{K \cdot S \to K'(S)}}{(K \cdot S) \cdot I \to K'(S) \cdot I} \qquad \frac{K'(S) \xrightarrow{I} S}{K'(S) \cdot I \to S}, \qquad \text{so} \quad (K \cdot S) \cdot I \to^{\star} S$$

[†] Abramsky, "The lazy λ-calculus", 1990.

Big-Step Semantics

- Notion of value: $v, w ::= I \mid K \mid S \mid K'(t) \mid S'(t) \mid S''(s, t)$
- Evaluation relation: $\Downarrow \subseteq \mathit{Terms} \times \mathit{Values}$
- Big-step rules:

$$\frac{}{v \Downarrow v} \qquad \frac{s \Downarrow I \quad t \Downarrow v}{s \cdot t \Downarrow v} \qquad \frac{s \Downarrow K'(r) \quad r \Downarrow v}{s \cdot t \Downarrow v}$$

$$\frac{s \Downarrow S''(r, q) \quad (r \cdot t)(q \cdot t) \Downarrow v}{s \cdot t \Downarrow v} \qquad \ldots$$

Equivalence Theorem: $t \Downarrow v \iff t \to^{\star} v \wedge v$ is a value

? How to prove it abstractly?

How to prove

$t \Downarrow v \iff t \rightarrow^{\star} v \wedge v$ is a value

abstractly?

Abstract Higher-Order GSOS

A Bit of Category Theory

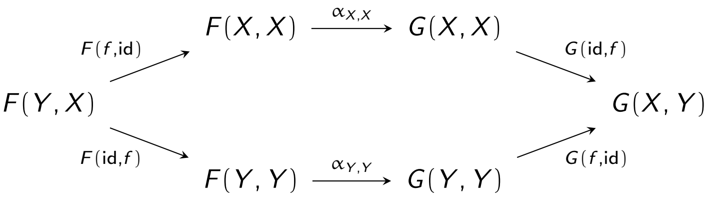
From the programming perspective:

- (Endo-)functor is a type constructor, e.g. $FX = X \times X$
- Natural transformation $\alpha\colon F \to G$ is a polymorphic function $\alpha_X\colon FX \to GX$, e.g. $\mathit{swap}\colon X \times X \to X \times X$
- Algebra is a map $a\colon FX \to X$, e.g. the free algebra of Σ-terms $\iota\colon \Sigma(\Sigma^\star X) \to \Sigma^\star X$ over variables X, and $\mu\Sigma \coloneqq \Sigma^\star\emptyset$
- Monad is such a functor T that morphisms $(f\colon X \to TY)_{X,Y\in\mathcal{C}}$ form a category, e.g. for $T = \mathcal{P}$ we obtain the category of relations

Dinaturality

Given two functors $F, G\colon \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathrm{D}$,

$\alpha = (\alpha_{X,Y}\colon F(X,Y) \to G(X,Y))_{X,Y\in\mathcal{C}}$ is a dinatural transformation if



commutes for any $f\colon X \to Y$

Example: evaluation transformation $\mathrm{ev}\colon C^X \times X \to C$

Higher-Order Abstract GSOS

A higher-order GSOS law in category $\mathcal{C}$ consists of

- Signature functor $\Sigma\colon \mathcal{C} \to \mathcal{C}$
- Behaviour functor $B\colon \mathcal{C}^{\mathsf{op}} \times \mathcal{C} \to \mathcal{C}$
- Family of maps $\rho_{X,Y}\colon \Sigma(X \times B(X,Y)) \to B(X, \Sigma^\star(X+Y))$ natural in Y and dinatural in X

For combinatory logic:

- $\mathcal{C}$ – category of sets
- $\Sigma X = \coprod_{f \in Ops} X^{arity(f)}$, $Ops = \{S, S', S'', K, K', I, \cdot\}$
- $B(X,Y) = Y^X + Y$
- ρ is induced by rules of operational semantics

Representing Rules

For $\mathcal{C} = \mathit{Set}$, $\Sigma X = \coprod_{f \in \mathit{Ops}} X^{\mathit{arity}(f)}$, $B(X, Y) = Y^X + Y$, HO-GSOS precisely correspond to sets of rules of the form[†]:

$$\frac{(x_j \to y_j)_{j \in W} \qquad (x_i \xrightarrow{z} y_i^z)_{i \in \overline{W},\, z \in \{x_1, \ldots, x_n\}}}{f(x_1, \ldots, x_n) \to t}$$

or

$$\frac{(x_j \to y_j)_{j \in W} \qquad (x_i \xrightarrow{z} y_i^z)_{i \in \overline{W},\, z \in \{x, x_1, \ldots, x_n\}}}{f(x_1, \ldots, x_n) \xrightarrow{x} t}$$

($W \subseteq \{1, \ldots, n\}$, $\overline{W} = \{1, \ldots, n\} \setminus W$)

Proof Idea: Yoneda-style argument

Generally, HO-GSOS vastly abstract this situation

† Goncharov, Milius, Schröder, Tsampas, and Urbat, “Towards a Higher-Order Mathematical Operational Semantics”, 2023.

Combinatory Logic as HO-GSOS

For example,

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

correspond to

$$\rho((p, p') \cdot (q, \_)) = p' \cdot q$$

$$\rho((p, f) \cdot (q, \_)) = f(q)$$

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

Operational Model

- Operational model $\gamma\colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$ abstracts derivability of one-step transitions $p \to p'$, $p \xrightarrow{t} p'$
- It is a unique solution to

$$\begin{array}{ccccc}
\Sigma(\mu\Sigma) & & \xrightarrow{\quad\iota\quad} & & \mu\Sigma \\
\Big\downarrow{\scriptstyle \Sigma\langle \mathrm{id},\gamma\rangle} & & & & \Big\downarrow{\scriptstyle \gamma} \\
\Sigma(\mu\Sigma \times B(\mu\Sigma,\mu\Sigma)) & \xrightarrow{\rho} & B(\mu\Sigma, \Sigma^\star(\mu\Sigma + \mu\Sigma)) & \xrightarrow{B(\mathrm{id},\nabla^\sharp)} & B(\mu\Sigma,\mu\Sigma)
\end{array}$$

- Alternatively: $\gamma = B(\mathrm{id}, \nabla^\sharp) \cdot \rho \cdot \Sigma\langle \mathrm{id}, \gamma\rangle \cdot \iota^{-1}$ (structural recursion)
- For combinatory logic: $\gamma(p) = p'$ iff $p \to p'$ and $\gamma(p) = f$ iff $\forall x.\, p \xrightarrow{x} f(x)$

Separation

Strict and Lazy Arguments

Rule

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

really means two rules

$$\frac{p \to p' \qquad q \to q'}{p \cdot q \to p' \cdot q} \qquad \frac{p \to p' \qquad (q \xrightarrow{r} q_r)_r}{p \cdot q \to p' \cdot q}$$

⚠ We cannot allow such ambivalence in big-step semantics:

$$\frac{s \Downarrow K'(r) \quad r \Downarrow v}{s \cdot t \Downarrow v}$$ behaves differently than $$\frac{s \Downarrow K'(r) \quad t \Downarrow w \quad r \Downarrow v}{s \cdot t \Downarrow v}$$

Separating example: $K'(I) \cdot \Omega$ (where $\Omega = (S \cdot I \cdot I) \cdot (S \cdot I \cdot I)$)

Solution: binary $\Sigma \colon \mathcal{C} \times \mathcal{C} \to \mathcal{C}$, for strict and lazy arguments

Separation

In $t \Downarrow v \iff t \to^\star v \wedge v$ is a value we need to define multistep semantics $\to^\star$ and values

Solution:

- Involve ω-continuous monad T, i.e such monad that morphisms $X \to T(Y+X)$ can be iterated. Examples $TX = X+1$, $TX = \mathcal{P}X$, $TX = D(X+1)$, $D =$ monad of probability distributions
- Assume separation $\Sigma(X, Y) = \Sigma^{v}(Y) + \Sigma^{c}(X, Y)$

Value signature — Computation signature

Separated Abstract HO-GSOS

Given $D\colon \mathcal{C}^{\mathsf{op}} \times \mathcal{C} \to \mathcal{C}$, $\Sigma_{\mathsf{v}}\colon \mathcal{C} \to \mathcal{C}$, $\Sigma_{\mathsf{c}}\colon \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ and a monad T, separated abstract HO-GSOS consists of

$$\rho^{\mathsf{v}}_X\colon \Sigma_{\mathsf{v}} X \to D(X, \Sigma^\star X)$$

$$\rho^{\mathsf{c}}_{X,Y}\colon \Sigma_{\mathsf{c}}(X \times (TD(X,Y) + TY), X) \to T\Sigma^\star(X+Y)$$

dinatural in X and natural in Y, and a distributive law

$$\chi_{X,Y}\colon \Sigma_{\mathsf{c}}(TX, Y) \to T\Sigma_{\mathsf{c}}(X,Y)$$

Abstract HO-GSOS can be recovered:

- $B(X,Y) = TD(X,Y) + TY, \quad \Sigma(X,Y) = \Sigma_{\mathsf{v}} Y + \Sigma_{\mathsf{c}}(X,Y)$
- $\rho = \eta \cdot D(\mathrm{id}, \Sigma^\star \mathrm{inl}) \cdot \rho^{\mathsf{v}} + \rho^{\mathsf{c}}$

Separated Abstract HO-GSOS: Properties

- Combinatory logic is separated (and so many others)
- Operational model $\beta\colon \mu\Sigma \to TD(\mu\Sigma, \mu\Sigma)$ gets separated to

$$\gamma^{\mathsf{v}}\colon \Sigma_{\mathsf{v}}\mu\Sigma \to TD(\mu\Sigma, \mu\Sigma) \qquad \gamma^{\mathsf{c}}\colon \Sigma_{\mathsf{c}}(\mu\Sigma, \mu\Sigma) \to T\mu\Sigma$$

  (**Slogan:** Values behave as values, computions as computations)

- We can define multi-step semantics $\beta\colon \mu\Sigma \to T(\Sigma^{\mathsf{v}}\mu\Sigma)$ as least fixpoint:

  Klesili lifting $\beta^{\sharp}\colon T\mu\Sigma \to TD(\mu\Sigma, \mu\Sigma)$

$$\beta = [\eta, \beta^{\sharp} \cdot \gamma^{\mathsf{c}}] \cdot \iota^{-1}$$

- Monad can be used for modelling other effects, e.g. add erratic choice with $p + q \to p$, $p + q \to q$, and $T = \mathcal{P}$

Abstract Big-Step SOS

Abstract Big-Step SOS

Abstract big-step SOS is a natural transformation

$$\xi\colon \Sigma_c(\Sigma_v X, X) \to T(\Sigma^\star X)$$

Assuming that $T = \mathrm{Id}$, this captures two kinds of rules:

$$\frac{}{g(p_1, \ldots, p_n) \Downarrow g(p_1, \ldots, p_n)} \quad (g \in \Sigma_v)$$

$$\frac{p_1 \Downarrow g_1(p_1^1, \ldots, p_{n_1}^1) \quad \ldots \quad p_k \Downarrow g_k(p_1^k, \ldots, p_{n_k}^k) \quad t \Downarrow v}{f(p_1, \ldots, p_k, \ldots, p_n) \Downarrow v} \quad (f \in \Sigma_c, g_i \in \Sigma_v)$$

where precisely k first arguments are strict

Abstract Big-Step SOS: Properties

- Big-step operational model is defined as least solution of simple recursive equation

$$\zeta = [\eta, \zeta^\sharp \cdot T\mu \cdot \xi^\sharp \cdot \chi \cdot \Sigma_c(\zeta, \mathrm{id})] \cdot \iota^{-1}$$

- This is neither structural, nor tail recursion, and so it must, e.g.

$$\frac{s \Downarrow S''(r, q) \quad (r \cdot t)(q \cdot t) \Downarrow v}{s \cdot t \Downarrow v}$$

- There is simple translation:

Separated Abstract HO-GSOS $\Rightarrow$ Abstract Big-Step SOS

(This is how big-step semantics of combinatory logic is obtained)

Separation isn't Enough

- Consider separated semantics

$$\frac{}{g(p) \xrightarrow{q} f(q)} \qquad \frac{}{\Omega \to \Omega} \qquad \frac{p \to p'}{f(p) \to g(p')} \qquad \frac{p \xrightarrow{p} p'}{f(p) \to p'}$$

- Only sensible big-step semantics is

$$\frac{}{g(p) \Downarrow g(p)} \qquad \frac{p \Downarrow g(q) \qquad g(q) \Downarrow v}{f(p) \Downarrow v}$$

- However $t \Downarrow v \iff t \to^{\star} v \wedge v$ is a value fails (!):
  $f(f(g(\Omega))) \to g(g(\Omega))$, but $f(f(g(\Omega))) \Downarrow g(\Omega)$

Strong Separation

Strong separation condition (omitted) abstracts the following: if a rule has at least one premise of the form $x_k \to x'_k$ then the conclusion of the rule must be

$$f(x_1, \ldots, x_n, y_1, \ldots, y_m) \to f(x'_1, \ldots, x'_n, y_1, \ldots, y_m)$$

where either $x_i \to x'_i$ occurs in the premise, or else, the premise contains a labeled transition for x_i, in which case $x'_i = x_i$.

Example:

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

Non-Example:

$$\frac{p \to p'}{f(p) \to g(p')}$$

Main Result

Theorem: if (ρ^v, ρ^c, χ) is strongly separated, and abstract big-step SOS law (ξ, χ) is generated by it, then multi-step semantics and big-step operrational model agree:

$$\beta = \zeta$$

Example: Call-by-Value

Call-by-value combinatory logic: combinators as before, plus

$$\frac{t \to t'}{t \cdot s \to t' \cdot s}\ (a) \qquad \frac{t \xrightarrow{r} t' \quad s \to s'}{t \cdot s \to t \cdot s'}\ (b) \qquad \frac{t \xrightarrow{s} t' \quad s \xrightarrow{r} s'}{t \cdot s \to t'}\ (c)$$

But no rule

$$\frac{t \to t' \quad s \to s'}{t \cdot s \to t' \cdot s'}$$

Hence, no strong separation. Solution: replace (b)–(c) with

$$\frac{s \xrightarrow{r} s'}{s \cdot t \to s \mathbin{◐} t} \qquad \frac{s \to s'}{t \mathbin{◐} s \to t \mathbin{◐} s'}$$

$$\frac{t \xrightarrow{r} t'}{s \mathbin{◐} t \to s \bullet t} \qquad \frac{t \to t'}{t \bullet s \to t' \bullet s} \qquad \frac{t \xrightarrow{s} t'}{t \bullet s \to t'}$$

This produces "pretty-big-step semantics"†

† Charguéraud, "Pretty-Big-Step Semantics", 2013.

Languages with Binders

- Small-step rules:

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

- Big-step rules:

$$\frac{}{\lambda x.\, p \Downarrow \lambda x.\, p} \qquad \frac{p \Downarrow \lambda x.\, p' \quad p'[q/x] \Downarrow v}{pq \Downarrow v}$$

We need to decompose (β) to

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

So, $p[q/x] = p'$ becomes new kind of transitions (!)

Space of substitution actions

Solution: Upgrade ρ^{v} to $\rho^{v}_{X,Y} : \Sigma_v(X \times (X \multimap Y)) \to D(X, \Sigma^\star(X + Y))$

use a presheave category as $\mathcal{C}$, for modeling languages with binders†

† Fiore, Plotkin, and Turi, "Abstract Syntax and Variable Binding", 1999.

Conclusions

- Abstract notions of small-step/big-step semantics
- A general and abstract $t \Downarrow v \iff t \to^{\star} v \wedge v$ is a value
- Functional implementation (in Haskell)

Further Work:

- Implementing proofs in proof assistant (WIP Meta-Semantics Agda Library[†])
- Cost semantics, probabilistic semantics (by varying T)
- Stateful semantics
- Other uses of (strong) separation (compositionality of observational equivalences?)

† https://github.com/sergey-goncharov/agda-meta-semantics

Thank You for Your Attention!

Higher-Order Abstract GSOS

Categorical Framework for Higher-Order Operational Semantics

Language

Signature $\cong$ Endofunctor $\Sigma\colon \mathbf{C} \to \mathbf{C}$ on a category $\mathbf{C}$, e.g.:

- $\mathbf{C} = \mathbf{Set}$, $\Sigma = \{0/0, a_i/0, +/2, \cdot/2\}$
- $\mathbf{C}$ = "nominal sets", $\Sigma X = \mathbb{A} + [\mathbb{A}]X + X \times X$

Behaviour

Behaviour = Mixed-variance functor $B\colon \mathbf{C}^{\mathrm{op}} \times \mathbf{C} \to \mathbf{C}$, e.g.:

- $B(X,Y) = Y^X + Y$ (deterministic)
- $B(X,Y) = \mathscr{P}_\omega(Y^X + Y)$ (non-deterministic)

HO Specification in GSOS Format

$$\frac{t \xrightarrow{a} t' \quad s \xrightarrow{\bar{a}} s'}{t \| s \xrightarrow{\tau} t' \| s'} \qquad \frac{t \xrightarrow{s} t'}{t s \longrightarrow t'} \qquad \frac{}{(\lambda x. f) s \longrightarrow f[s/x]}$$

$\Longrightarrow$ **Distributive law** ρ of Σ over B

Higher-Order Bialgebraic Semantics

Transition semantics is a unique solution $\gamma\colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$:

$$\begin{array}{ccccc}
\Sigma\mu\Sigma & \xrightarrow{\iota} & \mu\Sigma & \xrightarrow{\gamma} & B(\mu\Sigma,\mu\Sigma) \\
\downarrow{\scriptstyle \Sigma\langle \mathrm{id},\gamma\rangle} & & & & \uparrow{\scriptstyle B(\mathrm{id},\hat{\iota})} \\
\Sigma(\mu\Sigma \times B(\mu\Sigma,\mu\Sigma)) & \xrightarrow{\rho} & B(\mu\Sigma, \Sigma^\star(\mu\Sigma + \mu\Sigma)) & \xrightarrow{B(\mathrm{id},\Sigma^\star\nabla)} & B(\mu\Sigma, \Sigma^\star\mu\Sigma)
\end{array}$$

Generic Strong Applicative Bisimulation

Coalgebraic notion of **strong applicative bisimilarity** $\sim$ on initial Σ-algebra $\mu\Sigma$ (=algebra of programs) as a pullback

$$\begin{array}{ccc}
\sim & \longrightarrow & \mu\Sigma \\
\downarrow & \lrcorner & \downarrow{\scriptstyle \mathrm{coit}\,\gamma} \\
\mu\Sigma & \xrightarrow{\mathrm{coit}\,\gamma} & \nu\gamma. B(\mu\Sigma, \gamma)
\end{array}$$

Central Result: Compositionality for Free

Under certain general assumptions, $\sim$ is a congruence

Separated Abstract HO-GSOS and Abstract BSSOS in Haskell

Free Functors, and Signatures

```haskell
data Free s x  = Res x | Cont (s (Free s x))
type Initial s = Free s Void

newtype Mrg s x = Mrg (s x x)
sigOp = Cont . Mrg

data SepSig' sv sc x y = SigV (sv y) | SigC (sc x y)
type SepSig sv sc       = Mrg (SepSig' sv sc)
```

Values, Computations, Behaviours

```
type InitialV sv sc = sv (Initial (SepSig sv sc))
type InitialC sv sc = sc (Initial (SepSig sv sc))
                         (Initial (SepSig sv sc))

data SepBeh d x y = BehV (d x y) | BehC y
```

SepHOGSOS Type Class

```haskell
class (MixFunctor d, Functor sv, Bifunctor sc) =>
    SepHOGSOS sv sc d where
  rhoV :: sv x -> d x (Free (SepSig sv sc) x)
  rhoC :: sc (x, SepBeh d x y) x
                -> Free (SepSig sv sc) (Either x y)
```

Operational Model

```haskell
gammaV :: InitialV sv sc ->
   d (Initial (SepSig sv sc)) (Initial (SepSig sv sc))
gammaV t = mvmap id join $ rhoV t

gammaC :: Proxy d ->
   InitialC sv sc -> Initial (SepSig sv sc)
gammaC (p :: Proxy d) t =
 (rhoC @_ @_ @d $ first (id &&& gamma) t) >>= nabla
    where
      nabla = either id id
      gamma (Cont (Mrg (SigV v))) = BehV $ gammaV v
      gamma (Cont (Mrg (SigC c))) = BehC $ gammaC p c
```

Multi-Step Semantics

```
beta :: (Functor sv, Bifunctor sc, MixFunctor d,
     SepHOGSOS sv sc d) =>
  Proxy d -> Initial (SepSig sv sc) -> InitialV sv sc

beta (p :: Proxy d) (Cont (Mrg (SigV v))) = v
beta (p :: Proxy d) (Cont (Mrg (SigC c))) =
        beta p (gammaC p c)
```

XCL Signature

```haskell
data XCLV x
  = S
  | K
  | I
  | S' x
  | K' x
  | S'' x x

data XCLC x y
  = Comp x y
```

XCL as SepHOGSOS

```haskell
instance SepHOGSOS XCLV XCLC (->) where
  rhoV S = sigOp . SigV . S' . Res
  rhoV K = sigOp . SigV . K' . Res
  rhoV I = Res
  rhoV (S' t) = sigOp . SigV . S'' (Res t) . Res
  rhoV (K' t) = const (Res t)
  rhoV (S'' t s) = \r -> sigOp $ SigC $ Comp
      (sigOp $ SigC $ Comp (Res t) (Res r))
      (sigOp $ SigC $ Comp (Res s) (Res r))

  rhoC (Comp (_, BehC s) r) =
      sigOp (SigC $ Comp (Res $ Right s) (Res $ Left r))
  rhoC (Comp (_, BehV f) r) = Res (Right $ f r)
```

BSSOS Type Class

```haskell
class (Functor sv, Bifunctor sc) => BSSOS d sv sc where
  xi :: sc (sv x) x -> Free (SepSig sv sc) x

  zeta' :: Initial (SepSig sv sc) -> InitialV sv sc
  zeta' (Cont (Mrg (SigV v))) = v
  zeta' (Cont (Mrg (SigC c))) = zeta' @d $ join $ xi @d
      $ first (zeta' @d) c

  zeta :: InitialC sv sc -> InitialV sv sc
  zeta = zeta' @d . sigOp . SigC
```

From SepHOGSOS to BSSOS

```haskell
instance (SepHOGSOS sv sc d) => BSSOS d sv sc where
  xi :: sc (sv x) x -> Free (SepSig sv sc) x
  xi t = rhoCV (bimap ((sigOp . SigV &&&
                              mx_second @d join . rhoV)
                             . fmap return)
                  return t)
            >>= nabla
  where nabla = either id id
```

References I

- Abramsky, S. "The lazy λ-calculus". In: *Research topics in Functional Programming*. Addison Wesley, 1990, pp. 65–117.
- Charguéraud, Arthur. "Pretty-Big-Step Semantics". In: *Programming Languages and Systems*. Ed. by Matthias Felleisen and Philippa Gardner. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 41–60.
- Fiore, Marcelo P., Gordon D. Plotkin, and Daniele Turi. "Abstract Syntax and Variable Binding". In: *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*. IEEE Computer Society, 1999, pp. 193–202. URL: `https://doi.org/10.1109/LICS.1999.782615`.

References II

- Goncharov, Sergey, Stefan Milius, Lutz Schröder, Stelios Tsampas, and Henning Urbat. "Towards a Higher-Order Mathematical Operational Semantics". In: Proc. ACM Program. Lang. 7 (2023), pp. 632–658. DOI: 10.1145/3571215.
- Goncharov, Sergey, Stefan Milius, Stelios Tsampas, and Henning Urbat. "Bialgebraic Reasoning on Higher-Order Program Equivalence". In: *LICS*. 2024, pp. 1–13.

References III

Goncharov, Sergey, Alessio Santamaria, Lutz Schröder, Stelios Tsampas, and Henning Urbat. "Logical Predicates in Higher-Order Mathematical Operational Semantics". In: *Foundations of Software Science and Computation Structures - 27th International Conference, FoSSaCS 2024, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2024, Luxembourg City, Luxembourg, April 6-11, 2024, Proceedings, Part II*. Ed. by Naoki Kobayashi and James Worrell. Vol. 14575. Lecture Notes in Computer Science. Springer, 2024, pp. 47–69. DOI: `10.1007/978-3-031-57231-9\_3`. URL: `https://doi.org/10.1007/978-3-031-57231-9\_3`.

References IV

Goncharov, Sergey, Stelios Tsampas, and Henning Urbat. "Abstract Operational Methods for Call-by-Push-Value". In: Proc. ACM Program. Lang. (2025). accepted.

Urbat, Henning, Stelios Tsampas, Sergey Goncharov, Stefan Milius, and Lutz Schröder. "Weak Similarity in Higher-Order Mathematical Operational Semantics". In: *LICS*. 2023, pp. 1–13. DOI: 10.1109/LICS56636.2023.10175706. URL: https://doi.org/10.1109/LICS56636.2023.10175706.