Higher-order Mathematical Operational Semantics aka Higher-order Abstract GSOS

Towards a unifying theory of operational methods/logical relations

Stelios Tsampas plus collaborators (mainly) from Erlangen

December 21, 2024

Friedrich-Alexander-Universität Erlangen-Nürnberg

Patterns in Formal (Operational) Methods

1. Prove property X for a specific language A (likewise for a relational property)

2. Prove property X for a class of languages A

Patterns in Formal (Operational) Methods

1. Prove property X for a specific language A (likewise for a relational property)
   i. Design an operational semantics

2. Prove property X for a class of languages A

Patterns in Formal (Operational) Methods

1. Prove property X for a specific language A (likewise for a relational property)
   i. Design an operational semantics
   ii. Pick a method (e.g. logical relations, Howe's method, etc.)

2. Prove property X for a <u>class</u> of languages A

Patterns in Formal (Operational) Methods

1. Prove property X for a specific language A (likewise for a relational property)
   i. Design an operational semantics
   ii. Pick a method (e.g. logical relations, Howe's method, etc.)
   iii. Implement from scratch
2. Prove property X for a <u>class</u> of languages A

Patterns in Formal (Operational) Methods

1. Prove property X for a specific language A (likewise for a relational property)
   i. Design an operational semantics
   ii. Pick a method (e.g. logical relations, Howe's method, etc.)
   iii. Implement from scratch
2. Prove property X for a <u>class</u> of languages A
   i. Pick a simple representative in A

Patterns in Formal (Operational) Methods

1. Prove property X for a specific language A (likewise for a relational property)
   i. Design an operational semantics
   ii. Pick a method (e.g. logical relations, Howe's method, etc.)
   iii. Implement from scratch
2. Prove property X for a <u>class</u> of languages A
   i. Pick a simple representative in A
   ii. Design an operational semantics

Patterns in Formal (Operational) Methods

1. Prove property X for a specific language A (likewise for a relational property)
   i. Design an operational semantics
   ii. Pick a method (e.g. logical relations, Howe's method, etc.)
   iii. Implement from scratch
2. Prove property X for a class of languages A
   i. Pick a simple representative in A
   ii. Design an operational semantics
   iii. Implement from scratch

Patterns in Formal (Operational) Methods

1. Prove property X for a specific language A (likewise for a relational property)
    i. Design an operational semantics
    ii. Pick a method (e.g. logical relations, Howe's method, etc.)
    iii. Implement from scratch
2. Prove property X for a class of languages A
    i. Pick a simple representative in A
    ii. Design an operational semantics
    iii. Implement from scratch
    iv. Argue informally that this is a deep, insightful solution

Patterns in Formal (Operational) Methods

1. Prove property X for a specific language A (likewise for a relational property)
   i. Design an operational semantics
   ii. Pick a method (e.g. logical relations, Howe's method, etc.)
   iii. **Implement from scratch (hard, empirical, time-consuming)**
2. Prove property X for a <u>class</u> of languages A
   i. Pick a simple representative in A
   ii. Design an operational semantics
   iii. **Implement from scratch (hard, empirical, time-consuming)**
   iv. **Argue informally that this is a deep, insightful solution**

Past and future

- Operational methods, especially logical relations, has seen tremendous growth and successes in the past 30-odd years.
- Bright future ahead, no doubt, but...

Past and future

- Operational methods, especially logical relations, has seen tremendous growth and successes in the past 30-odd years.
- Bright future ahead, no doubt, but...

Past and future

- Operational methods, especially logical relations, has seen tremendous growth and successes in the past 30-odd years.
- Bright future ahead, no doubt, but...
- Can we make improve their **scaling** and stop **reinventing the wheel**?

Past and future

- Operational methods, especially logical relations, has seen tremendous growth and successes in the past 30-odd years.
- Bright future ahead, no doubt, but...
- Can we make improve their **scaling** and stop **reinventing the wheel**?

Systematize

Past and future

- Operational methods, especially logical relations, has seen tremendous growth and successes in the past 30-odd years.
- Bright future ahead, no doubt, but...
- Can we make improve their **scaling** and stop **reinventing the wheel**?

Generalize

Systematize

Past and future

- Operational methods, especially logical relations, has seen tremendous growth and successes in the past 30-odd years.
- Bright future ahead, no doubt, but...
- Can we make improve their **scaling** and stop **reinventing the wheel**?

Generalize

Systematize

Unify

MOS, aka Abstract GSOS, an exercise in composing mathematical concepts

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

Structural Op. Sem.

$$\begin{tikzcd}
\Sigma X \arrow[r, "g"] \arrow[d, "\Sigma(\mathrm{id}, h)"] & X \arrow[r, "h"] & BX \\
\Sigma(X \times BX) \arrow[rr, "\rho_X"] & & B(\Sigma^\star X) \arrow[u, "B(g^\star)"']
\end{tikzcd}$$

Bialgebras

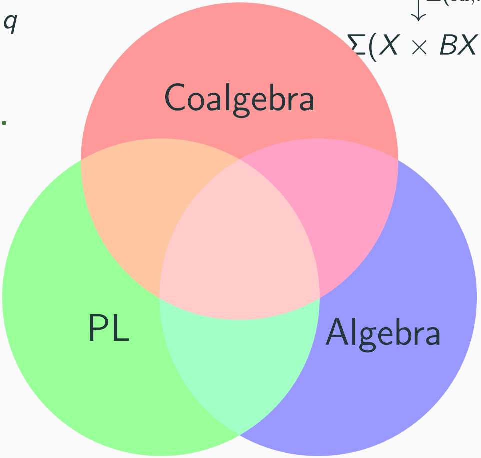
MOS, aka Abstract GSOS, an exercise in composing mathematical concepts

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

Structural Op. Sem.

$$\begin{tikzcd}
\Sigma X \arrow[r, "g"] \arrow[d, "\Sigma(\mathrm{id}, h)"] & X \arrow[r, "h"] & BX \\
\Sigma(X \times BX) \arrow[rr, "\rho_X"] & & B(\Sigma^{\star} X) \arrow[u, "B(g^{\star})"']
\end{tikzcd}$$

Bialgebras



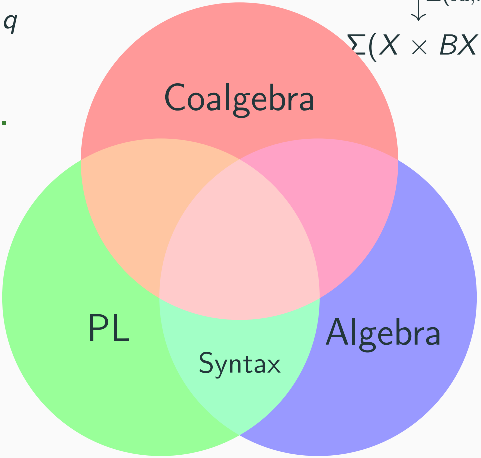
MOS, aka Abstract GSOS, an exercise in composing mathematical concepts

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

Structural Op. Sem.

$$\begin{tikzcd}
\Sigma X \arrow[r, "g"] \arrow[d, "\Sigma(\mathrm{id}, h)"] & X \arrow[r, "h"] & BX \\
\Sigma(X \times BX) \arrow[rr, "\rho_X"] & & B(\Sigma^\star X) \arrow[u, "B(g^\star)"']
\end{tikzcd}$$

Bialgebras



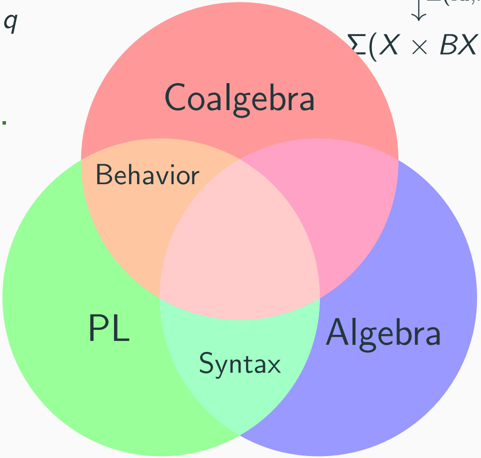
MOS, aka Abstract GSOS, an exercise in composing mathematical concepts

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

Structural Op. Sem.

$$\begin{tikzcd}
\Sigma X \arrow[r, "g"] \arrow[d, "\Sigma(\mathrm{id},h)"] & X \arrow[r, "h"] & BX \\
\Sigma(X \times BX) \arrow[rr, "\rho_X"] & & B(\Sigma^\star X) \arrow[u, "B(g^\star)"']
\end{tikzcd}$$

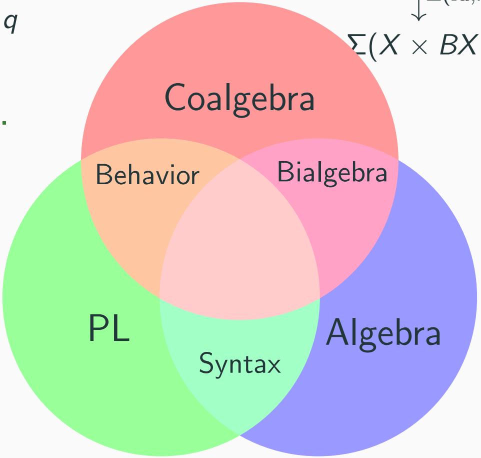
Bialgebras



MOS, aka Abstract GSOS, an exercise in composing mathematical concepts

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

Structural Op. Sem.



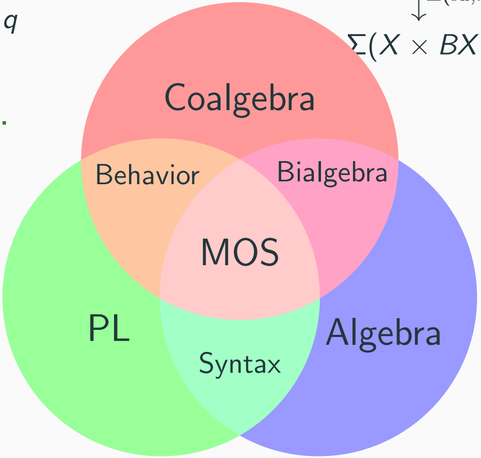
$$\begin{tikzcd}
\Sigma X \arrow[r, "g"] \arrow[d, "\Sigma(\mathrm{id}, h)"] & X \arrow[r, "h"] & BX \\
\Sigma(X \times BX) \arrow[rr, "\rho_X"] & & B(\Sigma^\star X) \arrow[u, "B(g^\star)"']
\end{tikzcd}$$

Bialgebras

MOS, aka Abstract GSOS, an exercise in composing mathematical concepts

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

Structural Op. Sem.



$$\begin{tikzcd}
\Sigma X \arrow[r, "g"] \arrow[d, "\Sigma(\mathrm{id}, h)"] & X \arrow[r, "h"] & BX \\
\Sigma(X \times BX) \arrow[rr, "\rho_X"] & & B(\Sigma^\star X) \arrow[u, "B(g^\star)"']
\end{tikzcd}$$

Bialgebras

"Why have I never heard of this?"

Pros/Cons

- ☺ Precise, elegant modelling of operational semantics
- ☺ Great with (first-order) process calculi [1]–[4]
- ☺ Effortless congruence results
- ☺ Allows the study of systems at a high level of generality

“Why have I never heard of this?”

Pros/Cons

- ☺ Precise, elegant modelling of operational semantics
- ☺ Great with (first-order) process calculi [1]–[4]
- ☺ Effortless congruence results
- ☺ Allows the study of systems at a high level of generality

- ☹ No higher-order languages

“Why have I never heard of this?”

Pros/Cons

- ☺ Precise, elegant modelling of operational semantics
- ☺ Great with (first-order) process calculi [1]–[4]
- ☺ Effortless congruence results
- ☺ Allows the study of systems at a high level of generality

- ☹ No higher-order languages
- ☹ Poor with imperative languages

"Why have I never heard of this?"

Pros/Cons

- ☺ Precise, elegant modelling of operational semantics
- ☺ Great with (first-order) process calculi [1]–[4]
- ☺ Effortless congruence results
- ☺ Allows the study of systems at a high level of generality

- ☹ No higher-order languages
- ☹ Poor with imperative languages
- ☹ What about weak bisimilarity, or $=^{\mathrm{ctx}}$?

“Why have I never heard of this?”

Pros/Cons

- ☺ Precise, elegant modelling of operational semantics
- ☺ Great with (first-order) process calculi [1]–[4]
- ☺ Effortless congruence results
- ☺ Allows the study of systems at a high level of generality

- ☹ No higher-order languages
- ☹ Poor with imperative languages
- ☹ What about weak bisimilarity, or $=^{\mathrm{ctx}}$?
- ☹ Categorical nonsense

“Why have I never heard of this?”

Pros/Cons

- ☺ Precise, elegant modelling of operational semantics
- ☺ Great with (first-order) process calculi [1]–[4]
- ☺ Effortless congruence results
- ☺ Allows the study of systems at a high level of generality

- 😎 ~~No~~ Yes higher-order languages [5]

"Why have I never heard of this?"

Pros/Cons

- ☺ Precise, elegant modelling of operational semantics
- ☺ Great with (first-order) process calculi [1]–[4]
- ☺ Effortless congruence results
- ☺ Allows the study of systems at a high level of generality

- 😎 ~~No~~ Yes higher-order languages [5]
- 😎 ~~Poor~~ Great with imperative languages (in preparation)

“Why have I never heard of this?”

Pros/Cons

- ☺ Precise, elegant modelling of operational semantics
- ☺ Great with (first-order) process calculi [1]–[4]
- ☺ Effortless congruence results
- ☺ Allows the study of systems at a high level of generality

- 😎 ~~No~~ Yes higher-order languages [5]
- 😎 ~~Poor~~ Great with imperative languages (in preparation)
- 😎 Howe’s method [6], log. rel. ([7], [8]), weak bisim. [9]

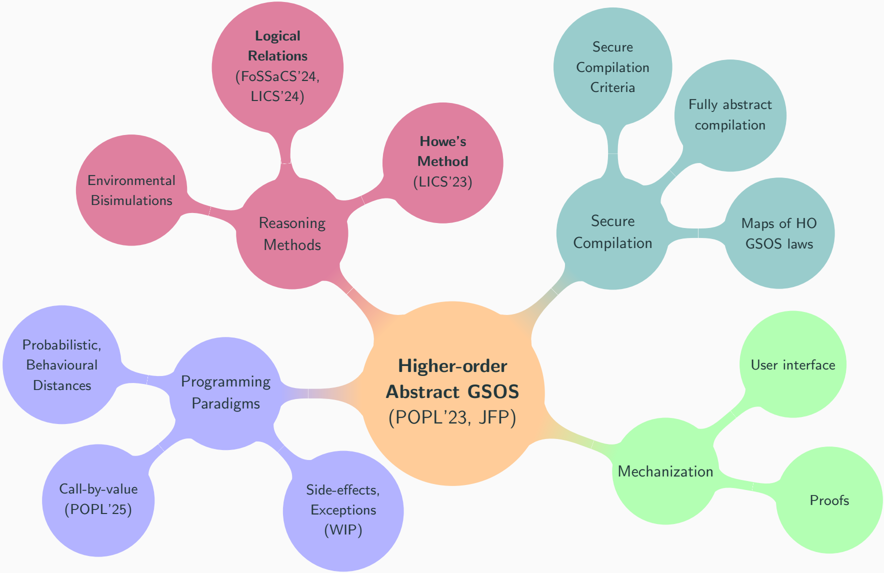
"Why have I never heard of this?"

Pros/Cons

- ☺ Precise, elegant modelling of operational semantics
- ☺ Great with (first-order) process calculi [1]–[4]
- ☺ Effortless congruence results
- ☺ Allows the study of systems at a high level of generality

- 😎 ~~No~~ Yes higher-order languages [5]
- 😎 ~~Poor~~ Great with imperative languages (in preparation)
- 😎 Howe's method [6], log. rel. ([7], [8]), weak bisim. [9]
- 😕 Categorical nonsense (for now)

Higher-Order Abstract GSOS, a research programme



Overview

HO-MOS or
Higher-order Abstract GSOS

Relational Reasoning

Abstract Reasoning with
Step-indexed Logical Relations

HO-MOS or Higher-order Abstract GSOS

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. $B = \mathcal{P}_{\mathsf{f}}^{\mathsf{L}}$).

(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:

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

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

From first-order to higher-order

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.

From first-order to higher-order

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 [1]

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

$$\Downarrow$$

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

From first-order to higher-order

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 [1]

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

$$\Downarrow$$

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

[1] That part was straightfoward, the modelling of the λ-calculus and the compositionality of the semantics, not so much ☺.

$\mathrm{SKI_u}$ 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: Operational semantics of the $\mathrm{SKI_u}$ calculus, our version of the SKI calculus, invented by Curry [10]. $\mathrm{SKI_u}$ in instance of an $\mathcal{HO}$ specification, a simple format of ours [5, §3].

$\mathrm{SKI_u}$ 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: Operational semantics of the $\mathrm{SKI_u}$ calculus, our version of the SKI calculus, invented by Curry [10]. $\mathrm{SKI_u}$ in instance of an $\mathcal{HO}$ specification, a simple format of ours [5, §3].

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

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

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!

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

combinator

application

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 $Y \in \mathcal{C}$.

Proposition

For every finitary signature $\check{\Sigma}$, 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 $B(X, Y) = Y + Y^X$.

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

$$\cong$$

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

Proposition

For every finitary signature $\check{\Sigma}$, 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 $B(X, Y) = Y + Y^X$.

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

$$\cong$$

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

Higher-Order Mathematical Operational Semantics

Proposition

For every finitary signature $\check{\Sigma}$, 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 $B(X, Y) = Y + Y^X$.

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

$$\cong$$

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

Higher-Order Mathematical Operational Semantics

Proposition

For every finitary signature $\check{\Sigma}$, 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 $B(X, Y) = Y + Y^X$.

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

$$\cong$$

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

Higher-Order Mathematical Operational Semantics

Proposition

For every finitary signature $\check{\Sigma}$, 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 $B(X, Y) = Y + Y^X$.

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

$$\cong$$

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

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.

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

Higher-Order Abstract GSOS

Operational rules

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

$\cong^\star$

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

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

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

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

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 **typed** combinator calculi, we have

$$\mathcal{C} = \mathsf{Set}^{\mathrm{Ty}} \quad \text{where } \mathrm{Ty} \text{ is the set of types}$$

$$\Sigma_\tau X = \coprod_{\sigma \in \mathrm{Ty}} X_{\sigma \multimap \tau} \times X_\sigma + \dots$$

$$B_{\sigma \multimap \tau}(X,Y) = Y_\tau + Y_\tau^{X_\sigma}$$

β-reduction or combinator

Higher-Order Abstract GSOS

Operational rules

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

$\cong^\star$

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 **typed** combinator calculi, we have

$$\mathcal{C} = \mathsf{Set}^{\mathrm{Ty}} \quad \text{where } \mathrm{Ty} \text{ is the set of types}$$

$$\Sigma_\tau X = \coprod_{\sigma \in \mathrm{Ty}} X_{\sigma \multimap \tau} \times X_\sigma + \dots$$

$$B_{\sigma \multimap \tau}(X,Y) = Y_\tau + Y_\tau^{X_\sigma}$$

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

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

$$\Sigma X = V + \delta X + X \times X \quad \text{(Fiore, Plotkin and Turi [11])}$$

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

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.

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

---

programs

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

  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 $\gamma : \underset{\text{programs}}{\mu\Sigma} \to B(\mu\Sigma, \mu\Sigma)$, ~~denotational model~~.

  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)

- $\underbrace{\textbf{Key feature: } \text{compositionality, i.e. bisimilarity is a congruence.}}_{\text{Proof: more complex than first-order case + needs mild assumptions.}}$

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 $t_1 \sim t_2$,

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

$+$ two symmetric conditions

Abstract modelling of Operational Semantics

Concrete/Abstract

Abstract modelling of Operational Semantics

Concrete/Abstract

1. Algebraic signature $\check{\Sigma}$

Abstract modelling of Operational Semantics

Concrete/Abstract

1. Algebraic signature $\check{\Sigma}$

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

Abstract modelling of Operational Semantics

Concrete/Abstract

1. Algebraic signature $\check{\Sigma}$
2. Program terms $\mu\Sigma$

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

Abstract modelling of Operational Semantics

Concrete/Abstract

1. Algebraic signature $\check{\Sigma}$
2. Program terms $\mu\Sigma$

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

Abstract modelling of Operational Semantics

Concrete/Abstract

1. Algebraic signature $\check{\Sigma}$
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 modelling of Operational Semantics

Concrete/Abstract

Concrete	Abstract
1. Algebraic signature $\check{\Sigma}$	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 $B\colon \mathcal{C}^{\mathsf{op}} \times \mathcal{C} \to \mathcal{C}$

Abstract modelling of Operational Semantics

Concrete/Abstract

1. Algebraic signature $\check{\Sigma}$
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 $B \colon \mathcal{C}^{\mathsf{op}} \times \mathcal{C} \to \mathcal{C}$

Abstract modelling of Operational Semantics

Concrete/Abstract

Concrete	Abstract
1. Algebraic signature $\check{\Sigma}$	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 $\dfrac{t \to t'}{t \cdot s \to t' \cdot s}$	4. Higher-order GSOS law $\rho_{X,Y}$

Abstract modelling of Operational Semantics

Concrete/Abstract

1. Algebraic signature $\check{\Sigma}$
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 modelling of Operational Semantics

Concrete/Abstract

Concrete	Abstract
1. Algebraic signature $\check{\Sigma}$	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 $\gamma\colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$

Abstract modelling of Operational Semantics

Concrete/Abstract

1. Algebraic signature $\check{\Sigma}$
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 modelling of Operational Semantics

Concrete/Abstract

1. Algebraic signature $\check{\Sigma}$
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 $\gamma \colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$
6. $B(\mu\Sigma, -)$-bisimulations

Abstract modelling of Operational Semantics

Concrete/Abstract

Concrete	Abstract
1. Algebraic signature $\check{\Sigma}$	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 modelling of Operational Semantics

Concrete/Abstract

1. Algebraic signature $\check{\Sigma}$
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 $\gamma\colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$
6. $B(\mu\Sigma, -)$-bisimulations

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

[5]: 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

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

We want to mode… …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 $\mathsf{Rel}(\mathcal{C})$ 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.

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

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 functors (and the rest of the constructions) to $\mathsf{Rel}(\mathcal{C})$ and $\mathsf{Pred}(\mathcal{C})$, a process that is known as relation (or predicate) lifting [12].

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

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 functors (and the rest of the constructions) to $\mathsf{Rel}(\mathcal{C})$ and $\mathsf{Pred}(\mathcal{C})$, a process that is known as relation (or predicate) lifting [12].

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

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 $t : \tau_1 \rightarrow \tau_2$, $s : \tau_1$ such that $P_{\tau_1 \rightarrow \tau_2}(t)$ 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 <u>canonical</u> 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 $t_1, t_2 : \tau_1 \rightarrow \tau_2$, $s_1, s_2 : \tau_1$ such that $R_{\tau_1 \rightarrow \tau_2}(t_1, t_2)$ 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.

Simple go-to example (untyped syntax this time)

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

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

Act II, Bisimulations. Prelude.

Simple go-to example (untyped syntax this time)

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

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

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

Act II, Bisimulations. Prelude.

Simple go-to example (untyped syntax this time)

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

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

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

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. Logical Relations.

Let R be a relation on the state space of a coalgebra $h : X \to B(X, X)$. We say that R is a logical relation (for h, h) if

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

Let R be a relation on the state space of a coalgebra $h : X \to B(X, X)$. We say that R is a logical relation (for h, h) if

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

Instantiate on $\gamma : \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$. A relation R is logical if the following hold for all t, s with $R(t, s)$:

1. If $t = \lambda x.t'$, then $s = \lambda x.s'$ and

   for all terms e_1, e_2 with $R(e_1, e_2)$, we have $R(t'[e_1/x], s'[e_2/x])$.

2. If $t \to t'$ then $s \to s'$ and $R(t', s')$.

Act II, Bisimulations. Logical Relations.

Let R be a relation on the state space of a coalgebra $h : X \to B(X, X)$. We say that R is a logical relation (for h, h) if

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

If R is logical, then the following is true for all $t, s \colon \sigma \rightarrowtriangle \tau$ with $R_{\sigma \rightarrowtriangle \tau}(t, s)$:

1. If $t = \lambda x \colon \sigma.t'$, then $s = \lambda x \colon \sigma.s'$ and

   $$\text{for all terms } e_1, e_2 \colon \sigma \text{ with } R_\sigma(e_1, e_2), \text{ we have } R_\tau(t'[e_1/x], s'[e_2/x]).$$

2. If $t \to t'$ then $s \to s'$ and $R_{\sigma \rightarrowtriangle \tau}(t', s')$.

Bisimulations, logical relations and step-indexing [8]

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

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

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

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

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	

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

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

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

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
5. Fundamental Property of Logical Relations

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

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
5. Fundamental Property of Logical Relations
6. Constructing **that** Logical Relation, **the chosen one**

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

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	5. Generalized induction $\overline{\Sigma}(R) \leq \iota^\star[R] \implies \Delta \leq R$
6. Constructing **that** Logical Relation, **the chosen one**	6. Error 404 : `Abstract Construction Missing`

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
5. Fundamental Property of Logical Relations
6. Constructing **that** Logical Relation, **the chosen one**
7. What's in it for me?

1. $P \rightarrowtail \mu\Sigma$, $R \rightarrowtail \mu\Sigma \times \mu\Sigma$
2. Complete, well-powered cat. $\mathcal{C}$
3. $P \leq h^\star[\overline{B}(P, P)]$
4. $R \leq (h \times \tilde{h})^\star[\overline{B}(R, R)]$
5. Generalized induction $\overline{\Sigma}(R) \leq \iota^\star[R] \implies \Delta \leq R$
6. Error 404 : `Abstract Construction Missing`

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 $\overline{\Sigma}(R) \leq \iota^{\star}[R] \implies \Delta \leq R$
6. Constructing **that** Logical Relation, **the chosen one**	6. Error 404 : `Abstract Construction Missing`
7. What's in it for me?	7. ???

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 :).

Abstract Reasoning with Step-indexed Logical Relations

Constructing step-indexed logical relation, Coalgebraically

Let's systematize the construction of a step-indexed logical relation, in a language-independent manner.

Constructing step-indexed logical relation, Coalgebraically

Let's systematize the construction of a step-indexed logical relation, in a language-independent manner.

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 $(\,\Box^{\overline{B},c,\tilde{c},\alpha} R \rightarrowtail X \times X\,)_\alpha$ by transfinite induction (writing $\Box^\alpha$ for simplicity):

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

Constructing step-indexed logical relation, Coalgebraically

Let's systematize the construction of a step-indexed logical relation, in a language-independent manner.

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):

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

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

$$\mathcal{L}^0_\tau(\Gamma) = \top_\tau(\Gamma) = \{(t,s) \mid \Gamma \vdash t, s\colon \tau\}$$

$$\mathcal{L}^{\alpha+1}_\tau = \mathcal{L}^\alpha_\tau \cap \mathcal{S}_\tau(\mathcal{L}^\alpha, \mathcal{L}^\alpha) \cap \mathcal{E}_\tau(\mathcal{L}^\alpha) \cap \mathcal{V}_\tau(\mathcal{L}^\alpha, \mathcal{L}^\alpha)$$

$$\mathcal{L}^\alpha_\tau(\Gamma) = \bigcap_{\beta<\alpha} \mathcal{L}^\alpha_\tau(\Gamma) \quad \text{for limit ordinals } \alpha.$$

$$\mathcal{S}_\tau(\Gamma)(Q,R) = \{(t,s) \mid \text{for all } \Delta \text{ and } Q_{\Gamma(x)}(\Delta)(u_x, v_x)\ (x \in |\Gamma|), \text{ one has } R_\tau(\Delta)(t[\vec{u}], s[\vec{v}])\},$$

$$\mathcal{E}_\tau(\Gamma)(R) = \{(t,s) \mid \text{if } t \to t' \text{ then } \exists s'.\, s \Rightarrow s' \wedge R_\tau(\Gamma)(t',s')\},$$

$$\mathcal{V}_{\tau_1 \boxtimes \tau_2}(\Gamma)(Q,R) = \{(t,s) \mid \text{if } t = \mathsf{pair}_{\tau_1,\tau_2}(t_1,t_2) \text{ then } \exists s_1, s_2.\, s \Rightarrow \mathsf{pair}_{\tau_1,\tau_2}(s_1,s_2) \wedge R_{\tau_1}(\Gamma)(t_1,s_1) \wedge R_{\tau_2}(\Gamma)(t_2,s_2)\},$$

$$\mathcal{V}_{\mu\alpha.\tau}(\Gamma)(Q,R) = \{(t,s) \mid \text{if } t = \mathsf{fold}_\tau(t') \text{ then } \exists s'.\, s \Rightarrow \mathsf{fold}_\tau(s') \wedge R_{\tau[\mu\alpha.\tau/\alpha]}(\Gamma)(t',s')\},$$

$$\mathcal{V}_{\tau_1 \multimap \tau_2}(\Gamma)(Q,R) = \{(t,s) \mid \text{for all } Q_{\tau_1}(\Gamma)(e,e'), \text{ if } t = \lambda x.t' \text{ then } \exists s'.\, s \Rightarrow \lambda x.s' \ \wedge\ R_{\tau_2}(\Gamma)(t'[e/x], s'[e'/x])\}.$$

Logical Relations, abstractly

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.

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

Logical Relations, abstractly

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.

Distilling the essence of Operational Methods

By systematizing Howe's method and (step-indexed) logical relations, we show that, assuming the operational semantics are sane in certain way [2], then

1. Howe's method can be applied.
2. The evident logical relation should be sound w.r.t. contextual equivalence.

[2] They form a HO-GSOS law and a lax bialgebra.

Thank you!

References

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.

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.

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.

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.

F. Bonchi, D. Petrisan, D. Pous, J. Rot, "Lax bialgebras and up-to techniques for weak bisimulations", in 26th International Conference on Concurrency Theory, CONCUR 2015, Madrid, Spain, September 1.4, 2015, L. Aceto, D. de Frutos-Escrig, Eds., ser. LIPIcs, vol. 42, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015, pp. 240–253, ISBN: 978-3-939897-91-0. DOI: 10.4230/LIPIcs.CONCUR.2015.240. [Online]. Available: http://www.dagstuhl.de/dagpub/978-3-939897-91-0.

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.

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.