Towards a Big-Step Higher-Order Mathematical Operational Semantics

Pouya Partow

Chair of Theoretical Computer Science
Friedrich-Alexander-Universität Erlangen-Nürnberg

Oberseminar Theoretische Informatik, June 2024

Overview

1. Reasoning on operational semantics

Overview

1. Reasoning on operational semantics
2. Rule Formats, GSOS, and HO-GSOS

Overview

1. Reasoning on operational semantics
2. Rule Formats, GSOS, and HO-GSOS
3. Bis-step rule format

Introduction: Operational semantics,

and our concern about it

Operational semantics is a way to formally specify how a programming language must behave.

Operational semantics is a way to formally specify how a programming language must behave.

Specification is given by a set of inference rules for reduction of well-formed terms.

Operational Semantics

Operational semantics is a way to formally specify how a programming language must behave.

Specification is given by a set of inference rules for reduction of well-formed terms.

It is often apposed to denotational semantics, roughly analogous to proof theory and model theory for a logic.

Operational Semantics

Operational semantics is a way to formally specify how a programming language must behave.

Specification is given by a set of inference rules for reduction of well-formed terms.

It is often apposed to denotational semantics, roughly analogous to proof theory and model theory for a logic.

It is suitable by nature to implement.

Example

xCL Combinatory logic: A modest (abstract) programming language.

Example

xCL Combinatory logic: A modest (abstract) programming language.

- Syntax:

$$\Lambda ::= I \mid K \mid K'(\Lambda) \mid S \mid S'(\Lambda) \mid S''(\Lambda, \Lambda) \mid \Lambda \circ \Lambda$$

Example

xCL Combinatory logic: A modest (abstract) programming language.

- Syntax:

$$\Lambda ::= I \mid K \mid K'(\Lambda) \mid S \mid S'(\Lambda) \mid S''(\Lambda, \Lambda) \mid \Lambda \circ \Lambda$$

- Operational semantics:

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

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

Example

In classic CL with β-reduction:

$$SKII \rightarrow_\beta (KI)(II) \rightarrow_\beta KII \rightarrow_\beta I$$

Or

$$SKII \rightarrow_\beta SI$$

Or ...

In classic CL with β-reduction:

$$SKII \rightarrow_\beta (KI)(II) \rightarrow_\beta KII \rightarrow_\beta I$$

Or

$$SKII \rightarrow_\beta SI$$

Or ...

In xCL:

$$SKII \rightarrow S'(K)II \rightarrow S''(K, I)I \rightarrow (KI)(II) \rightarrow K'(I)(II) \rightarrow I$$

Rigorous mathematical reasoning on programs is sometimes more favorable than empirical testing.

[1] Plotkin, "A Structural Approach to Operational Semantics", 1981

Rigorous mathematical reasoning on programs is sometimes more favorable than empirical testing.

It can be done by reasoning on operational meaning of the language that the program is written in[1].

[1] Plotkin, "A Structural Approach to Operational Semantics", 1981

Rigorous mathematical reasoning on programs is sometimes more favorable than empirical testing.

It can be done by reasoning on operational meaning of the language that the program is written in[1].

For example, operational semantics is suitable for reasoning about program equivalence.

[1] Plotkin, "A Structural Approach to Operational Semantics", 1981

Contexual Equivalence

Terms p and q are equivalent ($p \sim q$) iff for all contexts C

$$v \downarrow \quad \Rightarrow \quad C[p] \rightarrow^{\star} v \quad \Leftrightarrow \quad C[q] \rightarrow^{\star} v$$

Contexual Equivalence

Terms p and q are equivalent ($p \sim q$) iff for all contexts C

$$v \downarrow \quad \Rightarrow \quad C[p] \rightarrow^\star v \quad \Leftrightarrow \quad C[q] \rightarrow^\star v$$

It is a fundamental question that when the above equivalence hold (and therefore useful in many cases).

Contexual Equivalence

Terms p and q are equivalent ($p \sim q$) iff for all contexts C

$$v \downarrow \quad \Rightarrow \quad C[p] \rightarrow^{\star} v \quad \Leftrightarrow \quad C[q] \rightarrow^{\star} v$$

It is a fundamental question that when the above equivalence hold (and therefore useful in many cases).

Easier to check if we have proved the following property (compositionality):

$$p \sim q \Rightarrow C[p] \sim C[q]$$

Similar things that often happen in reasoning on operational semantics:

1. Doing these proofs are laborious tasks.

Stating a Problem

Similar things that often happen in reasoning on operational semantics:

1. Doing these proofs are laborious tasks.
2. Usually done by using sophisticated techniques (Logical Relations and Howe's Method).

Stating a Problem

Similar things that often happen in reasoning on operational semantics:

1. Doing these proofs are laborious tasks.
2. Usually done by using sophisticated techniques (Logical Relations and Howe's Method).
3. Proofs are often boilerplate.

Stating a Problem

Similar things that often happen in reasoning on operational semantics:

1. Doing these proofs are laborious tasks.
2. Usually done by using sophisticated techniques (Logical Relations and Howe's Method).
3. Proofs are often boilerplate.
4. Proofs are often brittle.

Stating a Problem

Similar things that often happen in reasoning on operational semantics:

1. Doing these proofs are laborious tasks.
2. Usually done by using sophisticated techniques (Logical Relations and Howe's Method).
3. Proofs are often boilerplate.
4. Proofs are often brittle.

The 3rd point looks somehow promising!

Similar things that often happen in reasoning on operational semantics:

1. Doing these proofs are laborious tasks.
2. Usually done by using sophisticated techniques (Logical Relations and Howe's Method).
3. Proofs are often boilerplate.
4. Proofs are often brittle.

The 3rd point looks somehow promising!

We may be able to prove more general statements that cover proofs for different cases.

Rule Formats, GSOS, and HO-GSOS

A rule format is a formal pattern for creating inference rules.

Rule Formats

A rule format is a formal pattern for creating inference rules.

They can also be seen like functions with parameters like, reducing terms, number of premises, ... to an specific set of inference rules.

Rule Formats

A rule format is a formal pattern for creating inference rules.

They can also be seen like functions with parameters like, reducing terms, number of premises, ... to an specific set of inference rules.

Maybe we can

1. create a sufficiently general rule format,

Rule Formats

A rule format is a formal pattern for creating inference rules.

They can also be seen like functions with parameters like, reducing terms, number of premises, ... to an specific set of inference rules.

Maybe we can

1. create a sufficiently general rule format,
2. prove important properties about it.

A rule format is a formal pattern for creating inference rules.

They can also be seen like functions with parameters like, reducing terms, number of premises, ... to an specific set of inference rules.

Maybe we can

1. create a sufficiently general rule format,
2. prove important properties about it.

Now one can define operational semantics (fit in the format) for a language with already proved features.

GSOS

One of these rule formats is called GSOS[2].

GSOS Rule Format

A rule is in GSOS format if it is in the following form:

$$\frac{\{x_i \xrightarrow{a} y_{ij}^{a}\}_{1\leq j\leq n_i^a}^{1\leq i\leq m, a\in A_i} \qquad \{x_i \stackrel{b}{\not\rightarrow}\}_{b\in B_i}^{1\leq i\leq m}}{f(x_1,\ldots,x_m) \xrightarrow{c} t}$$

[2] Bloom, Istrail, Meyer,"Bisimulation Can't be Traced", 1990

One of these rule formats is called GSOS[2].

GSOS Rule Format

A rule is in GSOS format if it is in the following form:

$$\frac{\{x_i \xrightarrow{a} y_{ij}^a\}_{1\leq j\leq n_i^a}^{1\leq i\leq m, a\in A_i} \qquad \{x_i \stackrel{b}{\not\rightarrow}\}_{b\in B_i}^{1\leq i\leq m}}{f(x_1,\ldots,x_m) \xrightarrow{c} t}$$

where:

- f is an operation of arity m,

[2] Bloom, Istrail, Meyer,"Bisimulation Can't be Traced", 1990

GSOS

One of these rule formats is called GSOS[2].

GSOS Rule Format

A rule is in GSOS format if it is in the following form:

$$\frac{\{x_i \xrightarrow{a} y_{ij}^a\}_{1\leq j\leq n_i^a}^{1\leq i\leq m, a\in A_i} \quad \{x_i \stackrel{b}{\not\rightarrow}\}_{b\in B_i}^{1\leq i\leq m}}{f(x_1,\ldots,x_m) \xrightarrow{c} t}$$

where:

- f is an operation of arity m,
- All x_i, y_{ij}^a are distinct variables,

[2] Bloom, Istrail, Meyer,"Bisimulation Can't be Traced", 1990

GSOS

One of these rule formats is called GSOS[2].

GSOS Rule Format

A rule is in GSOS format if it is in the following form:

$$\frac{\{x_i \xrightarrow{a} y_{ij}^a\}_{1\leq j\leq n_i^a}^{1\leq i\leq m, a\in A_i} \quad \{x_i \stackrel{b}{\not\rightarrow}\}_{b\in B_i}^{1\leq i\leq m}}{f(x_1,\ldots,x_m) \xrightarrow{c} t}$$

where:

- f is an operation of arity m,
- All x_i, y_{ij}^a are distinct variables,
- t is a term built over all the variables x_i, y_{ij}^a,

[2] Bloom, Istrail, Meyer,"Bisimulation Can't be Traced", 1990

One of these rule formats is called GSOS[2].

GSOS Rule Format

A rule is in GSOS format if it is in the following form:

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

where:

- f is an operation of arity m,
- All x_i, y^a_{ij} are distinct variables,
- t is a term built over all the variables x_i, y^a_{ij},
- There is a set L of labels containing c and including all the sets A_i and B_i.

[2] Bloom, Istrail, Meyer,"Bisimulation Can't be Traced", 1990

Rule formats seem to be intricate creatures.

Rule formats seem to be intricate creatures.

Category theory seems to be a suitable language to talk about rule formats.

Categorical Expression

Rule formats seem to be intricate creatures.

Category theory seems to be a suitable language to talk about rule formats.

Syntax of a language can be expressed by an endofunctor Σ.

Rule formats seem to be intricate creatures.

Category theory seems to be a suitable language to talk about rule formats.

Syntax of a language can be expressed by an endofunctor Σ. For example, for a set of operators of a language $\bar{\Sigma}$:

$\Sigma : Set \to Set, \Sigma X = \coprod_{f \in \bar{\Sigma}} X^{ar(f)}$

Rule formats seem to be intricate creatures.

Category theory seems to be a suitable language to talk about rule formats.

Syntax of a language can be expressed by an endofunctor Σ. For example, for a set of operators of a language $\bar{\Sigma}$:

$\Sigma : Set \to Set, \Sigma X = \coprod_{f \in \bar{\Sigma}} X^{ar(f)}$

Another functor extracted from the operational semantics named behavior functor B takes part in the categorical explanation.

Rule formats seem to be intricate creatures.

Category theory seems to be a suitable language to talk about rule formats.

Syntax of a language can be expressed by an endofunctor Σ. For example, for a set of operators of a language $\bar{\Sigma}$:

$\Sigma : Set \to Set, \Sigma X = \coprod_{f \in \bar{\Sigma}} X^{ar(f)}$

Another functor extracted from the operational semantics named behavior functor B takes part in the categorical explanation. For example:

$B : Set \to Set, BX = \mathcal{P}_{fin}(X)^L$

Categorical Explanation (Continue)

We call a set of GSOS rules a GSOS specification.

[3] Turi and Plotkin, "Towards Mathematical Operational Semantics", 1997

We call a set of GSOS rules a GSOS specification.

Every GSOS specification corresponds to a natural transformation ρ of the following form[3]:

$$\rho X = \coprod_{f \in \Sigma} (X \times \mathcal{P}_{fin}(X)^L)^{ar(f)} \to \mathcal{P}_{fin}(\Sigma^* X)^L$$

which is a categorical expression of a set consisting of rules of the following form:

$$\frac{\{x_i \xrightarrow{a} y_{ij}^a\}_{1 \le j \le n_i^a}^{1 \le i \le m, a \in A_i} \quad \{x_i \overset{b}{\not\to}\}_{b \in B_i}^{1 \le i \le m}}{f(x_1, \ldots, x_m) \xrightarrow{c} t}$$

[3] Turi and Plotkin, "Towards Mathematical Operational Semantics", 1997

GSOS specifications are instances of natural transformations of the following form called GSOS law:

$$\rho : \Sigma(Id \times B) \Rightarrow B\Sigma^*$$

GSOS specifications are instances of natural transformations of the following form called GSOS law:

$$\rho : \Sigma(Id \times B) \Rightarrow B\Sigma^*$$

GSOS law is parametric to two functors Σ and B.

Categorical Expression (Continue)

GSOS specifications are instances of natural transformations of the following form called GSOS law:

$$\rho : \Sigma(Id \times B) \Rightarrow B\Sigma^*$$

GSOS law is parametric to two functors Σ and B.

Different B can lead to creation of different rule formats!

GSOS specifications are instances of natural transformations of the following form called GSOS law:

$$\rho : \Sigma(Id \times B) \Rightarrow B\Sigma^*$$

GSOS law is parametric to two functors Σ and B.

Different B can lead to creation of different rule formats!

After setting B, the functor Σ can be set based on the language under study.

GSOS specifications are instances of natural transformations of the following form called GSOS law:

$$\rho : \Sigma(Id \times B) \Rightarrow B\Sigma^*$$

GSOS law is parametric to two functors Σ and B.

Different B can lead to creation of different rule formats!

After setting B, the functor Σ can be set based on the language under study.

Based on this, a categorical framework has been built to ease the reasoning on GSOS specifications.

Also the correspondences has lead to study of different flavors of GSOS rule formats.[4]

[4] Klin, "Bialgebras for structural operational semantics: An introduction", 2011

Shortcoming

Also the correspondences has lead to study of different flavors of GSOS rule formats.[4]

But it does not support "higher order behavior" i.e. passing programs as values to other programs.

[4] Klin, "Bialgebras for structural operational semantics: An introduction", 2011

Also the correspondences has lead to study of different flavors of GSOS rule formats.[4]

But it does not support "higher order behavior" i.e. passing programs as values to other programs.

For example:

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

[4] Klin, "Bialgebras for structural operational semantics: An introduction", 2011

Also the correspondences has lead to study of different flavors of GSOS rule formats.[4]

But it does not support "higher order behavior" i.e. passing programs as values to other programs.

For example:

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

Ditto for untyped λ-calculus.

[4] Klin, "Bialgebras for structural operational semantics: An introduction", 2011

HO-GSOS

Recently, the limitation has been solved by inventing HO-GSOS rule format[5].

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

Recently, the limitation has been solved by inventing HO-GSOS rule format[5].

The rule format is roughly the same:

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

And any term (instead of set of labels) can be a label for a transition.

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

HO-GSOS

Recently, the limitation has been solved by inventing HO-GSOS rule format[5].

The rule format is roughly the same:

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

And any term (instead of set of labels) can be a label for a transition.

And HO-GSOS law:

$$\rho(X, Y) = \Sigma(X \times B(X, Y)) \to B(X, \Sigma^*(X + Y))$$

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

HO-GSOS

Recently, the limitation has been solved by inventing HO-GSOS rule format[5].

The rule format is roughly the same:

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

And any term (instead of set of labels) can be a label for a transition.

And HO-GSOS law:

$$\rho(X, Y) = \Sigma(X \times B(X, Y)) \to B(X, \Sigma^*(X + Y))$$

And a categorical framework...

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

- Logical relations:
  - "Logical Predicates in Higher-Order Mathematical Operational Semantics", Goncharov, Santamaria, Schröder, Tsampas, and Urbat, 2024
  - "Bialgebraic Reasoning on Higher-Order Program Equivalence", Goncharov, Milius, Tsampas, Urbat, 2024
- Howe's method:
  - "Weak similarity in higher-order mathematical operational semantics", Urbat, Tsampas, Goncharov, Milius, Schröder, 2023

- Logical relations:
  - "Logical Predicates in Higher-Order Mathematical Operational Semantics", Goncharov, Santamaria, Schröder, Tsampas, and Urbat, 2024
  - "Bialgebraic Reasoning on Higher-Order Program Equivalence", Goncharov, Milius, Tsampas, Urbat, 2024
- Howe's method:
  - "Weak similarity in higher-order mathematical operational semantics", Urbat, Tsampas, Goncharov, Milius, Schröder, 2023

It is interesting! But still there are ways to make it better.

Among different ways to classify operational semantics, we have:

Small-step VS Big-step

Small-step vs Big-step

Among different ways to classify operational semantics, we have:

Small-step VS Big-step

Small-step describes every step.

Among different ways to classify operational semantics, we have:

Small-step VS Big-step

Small-step describes every step.

For a complete picture, steps must be composed.

Among different ways to classify operational semantics, we have:

Small-step VS Big-step

Small-step describes every step.

For a complete picture, steps must be composed.

Big-step describes all possible sequential reductions at once.

Small-step vs Big-step

Among different ways to classify operational semantics, we have:

Small-step VS Big-step

Small-step describes every step.

For a complete picture, steps must be composed.

Big-step describes all possible sequential reductions at once.

No judgment for divergent terms.

Small-step vs Big-step

Among different ways to classify operational semantics, we have:

Small-step VS Big-step

Small-step describes every step.

For a complete picture, steps must be composed.

Big-step describes all possible sequential reductions at once.

No judgment for divergent terms.

HO-GSOS is not suitable for expressing big-step semantics.

Among different ways to classify operational semantics, we have:

Small-step VS Big-step

Small-step describes every step.

For a complete picture, steps must be composed.

Big-step describes all possible sequential reductions at once.

No judgment for divergent terms.

HO-GSOS is not suitable for expressing big-step semantics.

Inability to specify transitively closed relations.

It is beneficial to have both small-step and big-step operational semantics for a language.

[6] Ciobaca, "From Small-step Semantics to Big-step semantics, Automatically", 2013

It is beneficial to have both small-step and big-step operational semantics for a language.

Creating small-step operational semantics from scratch is often easier.

[6] Ciobaca, "From Small-step Semantics to Big-step semantics, Automatically", 2013

It is beneficial to have both small-step and big-step operational semantics for a language.

Creating small-step operational semantics from scratch is often easier.

There has been a try[6] to create such automatic translation.

[6] Ciobaca, "From Small-step Semantics to Big-step semantics, Automatically", 2013

Translation from Small-step to Big-step

It is beneficial to have both small-step and big-step operational semantics for a language.

Creating small-step operational semantics from scratch is often easier.

There has been a try[6] to create such automatic translation.

But the assumptions highly limits the usability of the translation.

[6] Ciobaca, "From Small-step Semantics to Big-step semantics, Automatically", 2013

Translation from Small-step to Big-step

It is beneficial to have both small-step and big-step operational semantics for a language.

Creating small-step operational semantics from scratch is often easier.

There has been a try[6] to create such automatic translation.

But the assumptions highly limits the usability of the translation.

⋆ We are trying to make such a translation for operational semantics that fits in HO-GSOS.

[6] Ciobaca, "From Small-step Semantics to Big-step semantics, Automatically", 2013

Big-step Format

Our exploration for finding a good big-step format started with having *xCL* combinatory logic as a case study.

Starting Point: *xCL* Combinatory Logic

Our exploration for finding a good big-step format started with having *xCL* combinatory logic as a case study.

And we came up with this!:

$$\frac{}{v \Downarrow v} \qquad \frac{p \Downarrow I \quad q \Downarrow v}{pq \Downarrow v} \qquad \frac{p \Downarrow K}{pq \Downarrow K'(q)} \qquad \frac{p \Downarrow S}{pq \Downarrow S'(q)}$$

$$\frac{p \Downarrow K'(t) \quad t \Downarrow v}{pq \Downarrow v} \qquad \frac{p \Downarrow S'(t)}{pq \Downarrow S''(t, q)} \qquad \frac{p \Downarrow S''(s, t) \quad (sq)(tq) \Downarrow v}{pq \Downarrow v}$$

Example

We had $SKII \to^\star I$. Now, we have $SKII \Downarrow I$:

$$
\dfrac{
  \dfrac{\dfrac{\dfrac{}{S \Downarrow S}}{SK \Downarrow S'(K)}}{SKI \Downarrow S''(K, I)}
  \qquad
  \dfrac{\dfrac{\dfrac{}{K \Downarrow K}}{KI \Downarrow K'(I)} \quad \dfrac{}{I \Downarrow I}}{(KI)(II) \Downarrow I}
}{SKII \Downarrow I}
$$

Equivalence

We could prove the following:

Theorem

For all terms p and q in xCL combinatory logic, the following proposition holds:

$$p \to^{\star} q \quad \wedge \quad q \downarrow \qquad \Longleftrightarrow \qquad p \Downarrow q$$

We could prove the following:

Theorem

For all terms p and q in xCL combinatory logic, the following proposition holds:

$$p \to^\star q \quad \wedge \quad q\downarrow \qquad \Longleftrightarrow \qquad p \Downarrow q$$

Is not there something missing?!

We could prove the following:

Theorem

For all terms p and q in xCL combinatory logic, the following proposition holds:

$$p \to^\star q \quad \wedge \quad q\downarrow \qquad \Longleftrightarrow \qquad p \Downarrow q$$

Is not there something missing?!

What does $q\downarrow$ mean?

Equivalence

We could prove the following:

Theorem

For all terms p and q in xCL combinatory logic, the following proposition holds:

$$p \rightarrow^{\star} q \quad \wedge \quad q\downarrow \qquad \Longleftrightarrow \qquad p \Downarrow q$$

Is not there something missing?!

What does $q\downarrow$ mean? It is a value.

Equivalence

We could prove the following:

Theorem

For all terms p and q in xCL combinatory logic, the following proposition holds:

$$p \to^\star q \quad \wedge \quad q \downarrow \qquad \Longleftrightarrow \qquad p \Downarrow q$$

Is not there something missing?!

What does $q \downarrow$ mean? It is a value.

Values in xCL

All terms v of the following form are values in xCL:

$$v ::= I \mid K \mid S \mid K'(t) \mid S'(t) \mid S''(s,t)$$

Our goal is to generalize the result in the previous slide.

Our goal is to generalize the result in the previous slide.

How much general?

The Notion of Sound Translation

Our goal is to generalize the result in the previous slide.

How much general?

To the order of rule formats (instead of two operational semantics).

Our goal is to generalize the result in the previous slide.

How much general?

To the order of rule formats (instead of two operational semantics).

Sound Translation From Small-step to Big-step

For two rule formats $\mathcal{R}_1$ and $\mathcal{R}_2$ a sound translation from small-step to big-step is a construction that maps every specification in $\mathcal{R}_1$ to one of its equivalent big-step specifications in $\mathcal{R}_2$. Equivalent in the sense of the definition of equivalence in the previous slide.

Cool HO Specifications: Weakened HO-GSOS

Intermediate step: Cool HO specifications, a limited version of HO-GSOS.

Cool HO Specifications: Weakened HO-GSOS

Intermediate step: Cool HO specifications, a limited version of HO-GSOS.

Cool HO Specifications

A rule in the following formats in the cool HO specification format.

$$\frac{}{f(x_1, \ldots, x_m) \to t} \qquad \frac{}{g(y_1, \ldots, y_n) \xrightarrow{x} s}$$

$$\frac{x_j \to y_j}{f(x_1, \ldots, x_j, \ldots, x_m) \to f(x_1, \ldots, y_j, \ldots, x_m)}$$

$$\frac{(x_j \xrightarrow{x_k} x_j^k)_{k \in \{1,\ldots,m\}}}{f(x_1, \ldots, x_j, \ldots, x_m) \to t} \qquad \frac{(y_i \xrightarrow{y_l} y_i^l)_{l \in \{1,\ldots,n\}} \quad y_i \xrightarrow{x} y_i^x}{g(y_1, \ldots, y_i, \ldots, y_n) \xrightarrow{x} s}$$

Active vs Passive

In Cool HO specification we have this division for operators:

Active vs Passive

In Cool HO specification we have this division for operators:

Active and Passive Operators

An operator is called passive iff its reduction is specified by a rule without any premises.

In Cool HO specification we have this division for operators:

Active and Passive Operators

An operator is called passive iff its reduction is specified by a rule without any premises.

An operator is called active iff it is not passive.

Active vs Passive

In Cool HO specification we have this division for operators:

Active and Passive Operators

An operator is called passive iff its reduction is specified by a rule without any premises.

An operator is called active iff it is not passive.

Receiving Position

For an active operator f that its reduction is described in HO Specification, the receiving position of f is the position of its variable that reduces in the premise.

We consider this division for operators: $\Sigma = \Sigma^{\nu} \cup \Sigma^{\bar{\nu}}$, where $\Sigma^{\nu} \cap \Sigma^{\bar{\nu}} = \emptyset$

Computation vs Value

We consider this division for operators: $\Sigma = \Sigma^{\nu} \cup \Sigma^{\bar{\nu}}$, where $\Sigma^{\nu} \cap \Sigma^{\bar{\nu}} = \emptyset$

Ditto for terms (no matter closed or open).

Computation vs Value

We consider this division for operators: $\Sigma = \Sigma^{\nu} \cup \Sigma^{\bar{\nu}}$, where $\Sigma^{\nu} \cap \Sigma^{\bar{\nu}} = \emptyset$

Ditto for terms (no matter closed or open).

It surprisingly coincides with labeled/unlabeled reductions.

Computation terms only have unlabeled reduction, and value terms only have labeled reductions.

Computation vs Value

We consider this division for operators: $\Sigma = \Sigma^{v} \cup \Sigma^{\bar{v}}$, where $\Sigma^{v} \cap \Sigma^{\bar{v}} = \emptyset$

Ditto for terms (no matter closed or open).

It surprisingly coincides with labeled/unlabeled reductions.

Computation terms only have unlabeled reduction, and value terms only have labeled reductions. Like in the mentioned Cool example (xCL):

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

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

This is the big-step format that we are hopeful to find it equivalent with Cool HO specifications:

$$\frac{}{v \Downarrow v}$$

$$\frac{x_j \Downarrow g(y_1, \ldots, y_n) \quad t \Downarrow v}{f(x_1, \ldots, x_j, \ldots, x_m) \Downarrow v}$$

where j is the receiving position for f.

This is the big-step format that we are hopeful to find it equivalent with Cool HO specifications:

$$\frac{}{v \Downarrow v}$$

$$\frac{x_j \Downarrow g(y_1, \ldots, y_n) \quad t \Downarrow v}{f(x_1, \ldots, x_j, \ldots, x_m) \Downarrow v}$$

where j is the receiving position for f.

In all judgments $p \Downarrow v$, where $p \neq v$, p is a computation term, and v is a value term.

Construction (Under Construction!)

Inspired by our motivational example (*xCL*) our approach is to create a big-step rule for every pair consisting of one computation former and one value former.

Construction (Under Construction!)

Inspired by our motivational example (*xCL*) our approach is to create a big-step rule for every pair consisting of one computation former and one value former.

We have three cases for the pair (f, g), where $f \in \Sigma^{\bar{\nu}}$ and $g \in \Sigma^{\nu}$:

1. (f is active, g is passive)
2. (f is active, g is active)
3. (f is passive, g can be passive or active)

Construction (Continue)

For the first case:

$$\frac{x_j \to y_j}{f(x_1, \ldots, x_j, \ldots, x_m) \to f(x_1, \ldots, y_j, \ldots, x_m)} \quad \& \quad \frac{(x_j \xrightarrow{x_k} x_j^k)_{k \in \{1,\ldots,m\}}}{f(x_1, \ldots, x_j, \ldots, x_m) \to t}$$

$$\& \quad \frac{}{g(y_1, \ldots, y_n) \xrightarrow{x} s}$$

For the first case:

$$\frac{x_j \to y_j}{f(x_1, \ldots, x_j, \ldots, x_m) \to f(x_1, \ldots, y_j, \ldots, x_m)} \quad \& \quad \frac{(x_j \xrightarrow{x_k} x_j^k)_{k \in \{1,\ldots,m\}}}{f(x_1, \ldots, x_j, \ldots, x_m) \to t}$$

$$\& \quad \frac{}{g(y_1, \ldots, y_n) \xrightarrow{x} s}$$

$$\Downarrow$$

$$\frac{x_j \Downarrow g(y_1, \ldots y_n) \qquad t[g(\bar{y})/x_j, s[x_k/x]/x_j^k]_{k \in \{1,\ldots,m\}}[g(\bar{y})/x_j] \Downarrow v}{f(x_1, \ldots, x_j, \ldots, x_m) \Downarrow v}$$

For the first case:

$$\frac{x_j \to y_j}{f(x_1, \ldots, x_j, \ldots, x_m) \to f(x_1, \ldots, y_j, \ldots, x_m)} \quad \& \quad \frac{(x_j \xrightarrow{x_k} x_j^k)_{k \in \{1,\ldots,m\}}}{f(x_1, \ldots, x_j, \ldots, x_m) \to t}$$

$$\& \quad \frac{}{g(y_1, \ldots, y_n) \xrightarrow{x} s}$$

$$\Downarrow$$

$$\frac{x_j \Downarrow g(y_1, \ldots y_n) \qquad t[g(\bar{y})/x_j, s[x_k/x]/x_j^k]_{k \in \{1,\ldots,m\}}[g(\bar{y})/x_j] \Downarrow v}{f(x_1, \ldots, x_j, \ldots, x_m) \Downarrow v}$$

The second case is very similar to this case.

Some Explanation

It is like that we are taking one step of the reduction by using a rule, and letting other rules to handle the rest ($t[\ldots]$).

For

$$\frac{x_j \to y_j}{f(x_1, \ldots, x_j, \ldots, x_m) \to f(x_1, \ldots, y_j, \ldots, x_m)} \quad \& \quad \frac{(x_j \xrightarrow{x_k} x_j^k)_{k \in \{1,\ldots,m\}}}{f(x_1, \ldots, x_j, \ldots, x_m) \to t}$$

$$\& \quad \frac{}{g(y_1, \ldots, y_n) \xrightarrow{x} s}$$

we have

$$\frac{g(\bar{y}) \xrightarrow{g(\bar{y})} s[g(\bar{y})/x] \qquad (g(\bar{y}) \xrightarrow{x_k} s[x_k/x])_{k \in \{1,\ldots,m\} \setminus \{j\}}}{f(x_1, \ldots, g(\bar{y}), \ldots, x_m) \to t[g(\bar{y})/x_j, s[g(\bar{y})/x]/x_j^j, s[x_k/x]/x_j^k]_{k \in \{1,\ldots,m\} \setminus \{j\}}}$$

Example

$$\frac{}{Y \xrightarrow{t} t(Yt)} \qquad \frac{t \xrightarrow{t} s}{D(t) \rightarrow s} \qquad \frac{t \rightarrow t'}{D(t) \rightarrow D(t')}$$

$$\frac{p \rightarrow p'}{pq \rightarrow p'q} \qquad \frac{p \xrightarrow{q} p'}{pq \rightarrow p'}$$

$$\Downarrow$$

$$\frac{}{v \Downarrow v} \qquad \frac{p \Downarrow Y \quad p(Yp) \Downarrow v}{pq \Downarrow v} \qquad \frac{t \Downarrow Y \quad t(Yt) \Downarrow v}{D(t) \Downarrow v}$$

More Definitions (For the Third Case)

Active Term

We call an (open) term t active if

1. $t = x$, where x is a variable, or
2. $t = f(t_1, \ldots, t_n)$ with f being an active opertation whose receiving position is i and such that t_i is active again.

Active-Computation Term

We call an (open) term t active-computation if

1. $t = x$, where x is a variable, or
2. $t = f(t_1, \ldots, t_n)$ with f being an active-computation former whose receiving position is i and such that t_i is active-computation again.

Receiving Variable

Let t be an active term and let x be a variable that occurs in t precisely once. We call x a *receiving variable* for t if

1. $t = x$, or
2. $t = f(t_1, \ldots, t_n)$ and x is a receiving variable for t_i where i is the recieving position of f.

Another case division

Actually, we have already given the third case of construction only for passive computation formers that reduce to active terms, where we have these two mutually exclusive cases:

Actually, we have already given the third case of construction only for passive computation formers that reduce to active terms, where we have these two mutually exclusive cases:

1. t is an active-computation term.
2. t has the form $s[r/x]$ where r is a value and s is an active-computaition term, whose receiving variable is x.

Another case division

Actually, we have already given the third case of construction only for passive computation formers that reduce to active terms, where we have these two mutually exclusive cases:

1. t is an active-computation term.
2. t has the form $s[r/x]$ where r is a value and s is an active-computaition term, whose receiving variable is x.

We are hopeful that rules for operations that reduce to non-active terms, can be exchanged with rules for the operations in which they reduce to active or active-computation terms.

For the first subcase:

$$\frac{}{f(x_1,\ldots,x_m) \to t} \qquad \&$$

$$\left(\quad \frac{}{g(y_1,\ldots,y_n) \xrightarrow{x} s} \quad \oplus \quad \frac{(y_i \xrightarrow{y_l} y_i^l)_{l\in\{1,\ldots,n\}} \quad y_i \xrightarrow{x} y_i^x}{g(y_1,\ldots,y_i,\ldots,y_n) \xrightarrow{x} s} \quad \right)$$

For the first subcase:

$$\frac{}{f(x_1,\ldots,x_m)\to t} \qquad \&$$

$$\left(\frac{}{g(y_1,\ldots,y_n)\xrightarrow{x} s} \quad \oplus \quad \frac{(y_i \xrightarrow{y_l} y_i^l)_{l\in\{1,\ldots,n\}} \quad y_i \xrightarrow{x} y_i^x}{g(y_1,\ldots,y_i,\ldots,y_n)\xrightarrow{x} s} \right)$$

$$\Big\Downarrow (t = t'[x_j/x])$$

$$\frac{x_j \Downarrow g(y_1,\ldots,y_n) \qquad t'[g(y_1,\ldots,y_n)/x] \Downarrow v}{f(\bar{x}) \Downarrow v}$$

Further Steps

This is a perspective for further steps **IF** we succeed in the current step:

Further Steps

This is a perspective for further steps **IF** we succeed in the current step:

- Equivalent format for HO-GSOS (not limited to Cool).

Further Steps

This is a perspective for further steps **IF** we succeed in the current step:

- Equivalent format for HO-GSOS (not limited to Cool).
- Nondeterministic format.

Further Steps

This is a perspective for further steps **IF** we succeed in the current step:

- Equivalent format for HO-GSOS (not limited to Cool).
- Nondeterministic format.
- Categorical expression and framework.

Further Steps

This is a perspective for further steps **IF** we succeed in the current step:

- Equivalent format for HO-GSOS (not limited to Cool).
- Nondeterministic format.
- Categorical expression and framework.
- Translation in the opposite direction.

Further Steps

This is a perspective for further steps **IF** we succeed in the current step:

- Equivalent format for HO-GSOS (not limited to Cool).
- Nondeterministic format.
- Categorical expression and framework.
- Translation in the opposite direction.
- Labeled big-step transitions.

Further Steps

This is a perspective for further steps **IF** we succeed in the current step:

- Equivalent format for HO-GSOS (not limited to Cool).
- Nondeterministic format.
- Categorical expression and framework.
- Translation in the opposite direction.
- Labeled big-step transitions.
- Languages with variable-binders, like λ-calculus.

This is a perspective for further steps **IF** we succeed in the current step:

- Equivalent format for HO-GSOS (not limited to Cool).
- Nondeterministic format.
- Categorical expression and framework.
- Translation in the opposite direction.
- Labeled big-step transitions.
- Languages with variable-binders, like λ-calculus.
- Stateful semantics.

Vielen Dank! :-)