Operational Game Semantics for generative algebraic effects and handlers

(work in progress)

Hamza JAAFAR, Guilhem JABER

Nantes Universite, LS2N, INRIA Gallinette

January 14, 2024

Algebraic effects and handlers

- Impure behaviour given by operations on computations[1]
  (e.g choose for non-deterministic choice, raise for exceptions...)

[1] Gordon Plotkin and John Power. "Semantics for algebraic operations". In: *Electronic Notes in Theoretical Computer Science* 45 (2001), pp. 332–345.

Algebraic effects and handlers

- Impure behaviour given by operations on computations[1] (e.g choose for non-deterministic choice, raise for exceptions...)
- Impure behaviour is described by an equational theory on these operations

[1] Gordon Plotkin and John Power. "Semantics for algebraic operations". In: *Electronic Notes in Theoretical Computer Science* 45 (2001), pp. 332–345.

Algebraic effects and handlers

- Impure behaviour given by operations on computations[1]
  (e.g choose for non-deterministic choice, raise for exceptions...)
- Impure behaviour is described by an equational theory on these operations
- Account for monadic effects whose behaviour is independent of the current evaluation context

[1] Gordon Plotkin and John Power. "Semantics for algebraic operations". In: *Electronic Notes in Theoretical Computer Science* 45 (2001), pp. 332–345.

Algebraic effects and handlers

- Impure behaviour given by operations on computations[1] (e.g choose for non-deterministic choice, raise for exceptions...)
- Impure behaviour is described by an equational theory on these operations
- Account for monadic effects whose behaviour is independent of the current evaluation context

$$\mathrm{choose}(\mathcal{E}[\mathtt{M}], \mathcal{E}[\mathtt{N}]) \sim_{\mathsf{op}} \mathcal{E}[\mathrm{choose}(\mathtt{M}, \mathtt{N})]$$

[1] Gordon Plotkin and John Power. "Semantics for algebraic operations". In: *Electronic Notes in Theoretical Computer Science* 45 (2001), pp. 332–345.

Algebraic effects and handlers

- Impure behaviour given by operations on computations[1] (e.g choose for non-deterministic choice, raise for exceptions...)
- Impure behaviour is described by an equational theory on these operations
- Account for monadic effects whose behaviour is independent of the current evaluation context

$$\mathrm{choose}(\mathcal{E}[\mathtt{M}], \mathcal{E}[\mathtt{N}]) \sim_{\mathsf{op}} \mathcal{E}[\mathrm{choose}(\mathtt{M}, \mathtt{N})]$$

- Easier to structure compared to combining monadic effects.

[1] Gordon Plotkin and John Power. "Semantics for algebraic operations". In: *Electronic Notes in Theoretical Computer Science* 45 (2001), pp. 332–345.

Algebraic effects and handlers

- Impure behaviour given by operations on computations[1] (e.g choose for non-deterministic choice, raise for exceptions...)
- Impure behaviour is described by an equational theory on these operations
- Account for monadic effects whose behaviour is independent of the current evaluation context

$$\mathrm{choose}(\mathcal{E}[\mathtt{M}], \mathcal{E}[\mathtt{N}]) \sim_{\mathsf{op}} \mathcal{E}[\mathrm{choose}(\mathtt{M}, \mathtt{N})]$$

- Easier to structure compared to combining monadic effects.
- Handlers arise as homomorphisms between models of such algebraic theories.

[1] Gordon Plotkin and John Power. "Semantics for algebraic operations". In: *Electronic Notes in Theoretical Computer Science* 45 (2001), pp. 332–345.

Algebraic effects and Handlers, programmatically

- Effect operations are *constructors* or *producers* of effects.

Algebraic effects and Handlers, programmatically

- Effect operations are *constructors* or *producers* of effects.
- Handlers are *destructors* for effects.

Algebraic effects and Handlers, programmatically

- Effect operations are *constructors* or *producers* of effects.
- Handlers are *destructors* for effects.

A generalization of exception handlers (constructs such as **try** $\cdots$ **catch** or **try** $\cdots$ **with**) that can capture the *delimited continuation*.

Operations and effect handlers, concretely

Every operation symbol op comes with an arity

$$\mathbf{op} \quad : \quad \tau \to \sigma$$

Operations and effect handlers, concretely

Every operation symbol op comes with an arity

$$\mathbf{op} \quad : \quad \tau \to \sigma$$

- Performing an effect: **op** `V`

Operations and effect handlers, concretely

Every operation symbol op comes with an arity

$$\textbf{op} \quad : \quad \tau \to \sigma$$

- Performing an effect: **op** $\mathtt{V}$
- Handling an effect:

$$\begin{array}{lll} \mathtt{H} = & \{\textbf{return}\ x \mapsto \mathtt{N}\} & (\textit{return case}) \\ & \{\textbf{op}\ p\ \kappa \mapsto \mathtt{M}\} & (\text{op } \textit{case}) \end{array}$$

Operations and effect handlers, concretely

An effect $\mathtt{E}$ is typed by its signature $\Sigma_{\mathtt{E}} = \{(\mathbf{op_i} : \tau_i \to \sigma_i)_i\}$

Operations and effect handlers, concretely

An effect $\mathtt{E}$ is typed by its signature $\Sigma_{\mathtt{E}} = \{(\mathbf{op_i} : \tau_i \to \sigma_i)_i\}$

Example (Global state)

$$E^{\tau}_{state} = \{\mathbf{set} : \tau \to 1, \mathbf{get} : 1 \to \tau\}$$

Operations and effect handlers, concretely

An effect $\mathtt{E}$ is typed by its signature $\Sigma_{\mathtt{E}} = \{(\mathbf{op_i} : \tau_i \to \sigma_i)_i\}$

Example (Global state)

$$E^{\tau}_{state} = \{\mathbf{set} : \tau \to 1, \mathbf{get} : 1 \to \tau\}$$

What if we want multiple states holding values of the type τ.

Operations and effect handlers, concretely

An effect $\mathtt{E}$ is typed by its signature $\Sigma_{\mathtt{E}} = \{(\mathbf{op_i} : \tau_i \to \sigma_i)_i\}$

Example (Global state)

$$E^{\tau}_{state} = \{\mathbf{set} : \tau \to 1, \mathbf{get} : 1 \to \tau\}$$

What if we want multiple states holding values of the type τ.

Generally, how to deal with multiple occurences of the same effect type E without forefeiting modularity?

The Eff[2] approach

Use of a distinct identifier (names) ι for each instance of an effect E.

[2] Andrej Bauer and Matija Pretnar. "Programming with algebraic effects and handlers". In: *Journal of Logical and Algebraic Methods in Programming* 84.1 (2015). Special Issue: The 23rd Nordic Workshop on Programming Theory (NWPT 2011) Special Issue: Domains X, International workshop on Domain Theory and applications, Swansea, 5-7 September, 2011, pp. 108–123.

The Eff[2] approach

Use of a distinct identifier (names) ι for each instance of an effect $\texttt{E}$.

- Performing an effect: $\iota\#\mathsf{op}_{\texttt{E}}\ \texttt{V}$
- Handling an effect: $\texttt{H} = \{\iota\#\mathsf{op}_{\texttt{E}}\ p\ \kappa \mapsto \texttt{M}\}$ ($\iota\#\mathsf{op}_{\texttt{E}}$ case)

[2] Andrej Bauer and Matija Pretnar. "Programming with algebraic effects and handlers". In: *Journal of Logical and Algebraic Methods in Programming* 84.1 (2015). Special Issue: The 23rd Nordic Workshop on Programming Theory (NWPT 2011) Special Issue: Domains X, International workshop on Domain Theory and applications, Swansea, 5-7 September, 2011, pp. 108–123.

Programming Language

Syntax: Fine-grained call-by-value

Values	$\mathtt{V}, \mathtt{W}$	$\triangleq$	$x \mid \lambda x : \tau.\mathtt{M} \mid \iota$
Terms	$\mathtt{M}, \mathtt{N}$	$\triangleq$	$\textbf{return}\ \mathtt{V} \mid \mathtt{V}\ \mathtt{V} \mid \textbf{match}\ \mathtt{V}\ \textbf{with}\ (\mathtt{P}_i \to \mathtt{N}_i)_{i \in I}$
			$\mid \mathsf{let}\, x = \mathtt{M}\ \mathsf{in}\, \mathtt{N}$
			$\mid \mathtt{V\#op}\ \mathtt{W} \mid \textbf{handle}\ M\ \textbf{with}\ \mathtt{H}$
Handlers	$\mathtt{H}$	$\triangleq$	$\{\textbf{return}\ x \mapsto \mathtt{M}\} \mid \{\mathtt{V\#op}\ x\ \kappa \mapsto \mathtt{M}\} \uplus \mathtt{H}$
ECxts	$\mathcal{E}$	$\triangleq$	$\bullet \mid \mathsf{let}\, x = \mathcal{E}\ \mathsf{in}\, \mathtt{M} \mid \textbf{handle}\ \mathcal{E}\ \textbf{with}\ \mathtt{H}$

Dynamic generation of effects

New construct

Given an effect given by the type (signature) $\mathtt{E}$.

New construct: $$\mathtt{M}, \mathtt{N} \triangleq \cdots \mid \mathtt{new}\,\mathtt{E}$$

New construct

Given an effect given by the type (signature) $\mathtt{E}$.

New construct: $\mathtt{M}, \mathtt{N} \quad \triangleq \quad \cdots \mid \mathtt{new}\,\mathtt{E}$

Operational Semantics: $(\mathtt{new}\,\mathtt{E}; \mathcal{V}) \mapsto (\mathbf{return}\ \iota; \mathcal{V} \uplus \{\iota\})$

Disclosure and contextual equivalence

Consider the following variation of an example from[3]

$$f\ (\lambda x.5)$$

[3] Dariusz Biernacki et al. "Handle with care: relational interpretation of algebraic effects and handlers". In: *Proceedings of the ACM on Programming Languages* 2.POPL (2017), pp. 1–30.

Disclosure and contextual equivalence

Consider the following variation of an example from[3]

$$f\,(\lambda x.5)$$

$$\simeq_{ctx}$$

$$\begin{array}{l}
\texttt{let}\ y = \texttt{new}\,\texttt{E}\ \texttt{in} \\
\textbf{handle} \\
\quad f\,(\lambda x.y\#\texttt{op}\,()) \\
\textbf{with}\ \{\textbf{return}\ x \mapsto \textbf{return}\,x\} \\
\qquad\ \{y\#\texttt{op}\ x\ \kappa \mapsto \kappa\ 5\}
\end{array}$$

[3] Dariusz Biernacki et al. "Handle with care: relational interpretation of algebraic effects and handlers". In: *Proceedings of the ACM on Programming Languages* 2.POPL (2017), pp. 1–30.

Disclosure and contextual equivalence (cont.)

Now consider a variation of the previous example:

$$\text{let } y = \texttt{new E} \text{ in } g\ y; f\ (\lambda x.5)$$

Disclosure and contextual equivalence (cont.)

Now consider a variation of the previous example:

$$\text{let } y = \texttt{new}\,\texttt{E} \text{ in } g\ y; f\ (\lambda x.5)$$

$$\not\simeq_{ctx}$$

$$\begin{array}{l}
\texttt{let}\ y = \texttt{new}\,\texttt{E}\ \texttt{in} \\
\textbf{handle} \\
\quad g\ y\ ;\ f\ (\lambda x. y\#\texttt{op}\,())\\
\textbf{with}\ \{\textbf{return}\ x \mapsto \textbf{return}\,x\} \\
\qquad \{y\#\texttt{op}\ x\ \kappa \mapsto \kappa\ 5\}
\end{array}$$

Operational game semantics model

Existing fully-abstract models for effect handlers

Adaptation of Lassen's normal-form bisimulation[4]

[4] Dariusz Biernacki, Sergueï Lenglet, and Piotr Polesiuk. "A complete normal-form bisimilarity for algebraic effects and handlers". In: *Formal Structures for Computation and Deduction*. 2020.

Existing fully-abstract models for effect handlers

Adaptation of Lassen's normal-form bisimulation[4]

- Untyped calculus, global set of operations

[4] Dariusz Biernacki, Sergueï Lenglet, and Piotr Polesiuk. "A complete normal-form bisimilarity for algebraic effects and handlers". In: *Formal Structures for Computation and Deduction*. 2020.

Existing fully-abstract models for effect handlers

Adaptation of Lassen's normal-form bisimulation[4]

- Untyped calculus, global set of operations
- Completeness of the model does not rely on having additional stateful effect in the language.

[4] Dariusz Biernacki, Sergueï Lenglet, and Piotr Polesiuk. "A complete normal-form bisimilarity for algebraic effects and handlers". In: *Formal Structures for Computation and Deduction*. 2020.

Existing fully-abstract models for effect handlers

Adaptation of Lassen's normal-form bisimulation[4]

- Untyped calculus, global set of operations
- Completeness of the model does not rely on having additional stateful effect in the language.

[4] Dariusz Biernacki, Sergueï Lenglet, and Piotr Polesiuk. "A complete normal-form bisimilarity for algebraic effects and handlers". In: *Formal Structures for Computation and Deduction*. 2020.

Operational Game Semantics *OGS*

- *Trace semantics* following the operational evaluation of a program (Proponent) and tracing its interaction with its environment (Opponent).

Operational Game Semantics *OGS*

- *Trace semantics* following the operational evaluation of a program (Proponent) and tracing its interaction with its environment (Opponent).
- A trace is an alternating sequence of P-moves (noted with an overline) and O-moves, they can either be:

Operational Game Semantics *OGS*

- *Trace semantics* following the operational evaluation of a program (Proponent) and tracing its interaction with its environment (Opponent).
- A trace is an alternating sequence of P-moves (noted with an overline) and O-moves, they can either be:
  - Questions:

$$\overline{f}(\mathtt{A}, c) \qquad | \qquad f(\mathtt{A}, c)$$

    (requesting the result of f $\mathtt{A}$ as an answer in c)

Operational Game Semantics *OGS*

- *Trace semantics* following the operational evaluation of a program (Proponent) and tracing its interaction with its environment (Opponent).
- A trace is an alternating sequence of P-moves (noted with an overline) and O-moves, they can either be:
  - Questions:
    $$\overline{f}(\mathtt{A}, c) \qquad | \qquad f(\mathtt{A}, c)$$
    (requesting the result of $f\ \mathtt{A}$ as an answer in c)
  - Answers:
    $$\overline{c}(\mathtt{A}) \qquad | \qquad c(\mathtt{A})$$

Examples

Let's consider the trace of

$$f\ (\lambda x.5)$$

representing the interaction with the environment given by the evaluation context

$$\text{let } f = (\lambda g.g\ \texttt{V}; \textbf{return}\ \texttt{true}) \text{ in } []$$

Examples

Let's consider the trace of

$$f\ (\lambda x.5)$$

representing the interaction with the environment given by the evaluation context

$$\text{let } f = (\lambda g.g\ \mathtt{V}; \textbf{return}\ \mathtt{true}) \text{ in } []$$

yielding the trace

$$\overline{f}(g, c)$$

Examples

Let's consider the trace of

$$f\ (\lambda x.5)$$

representing the interaction with the environment given by the evaluation context

$$\text{let } f = (\lambda g.g\ \texttt{V}; \textbf{return}\ \texttt{true}) \text{ in } []$$

yielding the trace

$$\overline{f}(g, c)\ g(\texttt{A}, d)$$

Examples

Let's consider the trace of

$$f\ (\lambda x.5)$$

representing the interaction with the environment given by the evaluation context

$$\text{let } f = (\lambda g.g\ \mathtt{V}; \textbf{return}\ \mathtt{true}) \text{ in } []$$

yielding the trace

$$\overline{f}(g, c)\ g(\mathtt{A}, d)\ \overline{d}(5)$$

Examples

Let's consider the trace of

$$f\ (\lambda x.5)$$

representing the interaction with the environment given by the evaluation context

$$\text{let } f = (\lambda g.g\ \texttt{V}; \textbf{return}\ \texttt{true}) \text{ in } []$$

yielding the trace

$$\overline{f}(g, c)\ g(\texttt{A}, d)\ \overline{d}(5)\ c(\texttt{true})$$

Operational Game Semantics (cont.)

Normal Forms:

$$\mathbf{M_{nf}} = \mathcal{E}[f\ V] \mid \mathbf{return}\ V$$

Operational Game Semantics (cont.)

Normal Forms:

$$\mathbf{M_{nf}} = \mathcal{E}[f\ V] \mid \textbf{return}\ V$$

- $\mathcal{E}[f\ V]$ calls for a P-question of the shape $\overline{f}(\mathtt{A},\ c)$

Operational Game Semantics (cont.)

Normal Forms:

$$\mathbf{M_{nf}} = \mathcal{E}[f\ V] \mid \textbf{return}\ V$$

- $\mathcal{E}[f\ V]$ calls for a P-question of the shape $\overline{f}(\mathtt{A}, c)$
- **return** V calls for an answer of the shape $\overline{c}(\mathtt{A})$

Operational Game Semantics (cont.)

Normal Forms:

$$\mathbf{M_{nf}} = \mathcal{E}[f\ V] \mid \textbf{return}\ V$$

- $\mathcal{E}[f\ V]$ calls for a P-question of the shape $\overline{f}(\mathtt{A}, c)$
- **return** V calls for an answer of the shape $\overline{c}(\mathtt{A})$

The denotation $[\![\mathtt{M}]\!]_{\mathsf{ogs}}$ of a given program $\mathtt{M}$ is the set of all possible traces generated by $\mathtt{M}$

OGS model for algebraic effects and handlers

Algebraic effects introduce new normal forms:

$$\mathbf{M_{nf}} = \cdots \mid \mathcal{E}[\iota\#\mathit{op}\, V] \qquad \text{when } \iota\#\mathsf{op} \notin \mathrm{hdl}(\mathcal{E})$$

OGS model for algebraic effects and handlers

Algebraic effects introduce new normal forms:

$$\mathbf{M_{nf}} = \cdots \mid \mathcal{E}[\iota \# op\, V] \qquad \text{when } \iota \# \mathtt{op} \notin \mathrm{hdl}(\mathcal{E})$$

Extending the interaction interface with new moves that account for *observable* effectful operations.

OGS model for algebraic effects and handlers

Algebraic effects introduce new normal forms:

$$\mathbf{M_{nf}} = \cdots \mid \mathcal{E}[\iota \# op\, V] \qquad \text{when } \iota\#\mathtt{op} \notin \mathrm{hdl}(\mathcal{E})$$

Extending the interaction interface with new moves that account for *observable* effectful operations.

But, what counts as *observable*?

Accomodating the *OGS* model for effect name disclosure

When the program performs an effect

$$\iota\#\texttt{op V}$$

Accomodating the *OGS* model for effect name disclosure

When the program performs an effect

$$\iota\#\mathtt{op}\ \mathtt{V}$$

- Public: Opponent could potentially handle the effect.

Accomodating the *OGS* model for effect name disclosure

When the program performs an effect

$$\iota\#\texttt{op V}$$

- Public: Opponent could potentially handle the effect.
- Private: Opponent can only forward the effect to any enclosing Player's handling context.

Accomodating the *OGS* model for effect name disclosure

Algebraic effects introduce new normal forms:

$$\mathbf{M_{nf}} = \cdots \mid \mathcal{E}[\iota\#\mathit{op}\; V] \qquad \text{when } \iota\#\mathtt{op} \notin \mathrm{hdl}(\mathcal{E})$$

Accomodating the *OGS* model for effect name disclosure

Algebraic effects introduce new normal forms:

$$\mathbf{M_{nf}} = \cdots \mid \mathcal{E}[\iota\#op\ V] \qquad \text{when } \iota\#\mathsf{op} \notin \mathrm{hdl}(\mathcal{E})$$

- observable effect move: $\overline{c}[\iota\#\mathsf{op}\ \mathtt{A}\ \kappa]$

Accomodating the *OGS* model for effect name disclosure

Algebraic effects introduce new normal forms:

$$\mathbf{M_{nf}} = \cdots \mid \mathcal{E}[\iota\#op\ V] \qquad \text{when } \iota\#\mathrm{op} \notin \mathrm{hdl}(\mathcal{E})$$

- observable effect move: $\overline{c}[\iota\#\mathrm{op}\ \mathtt{A}\ \kappa]$
- private effect: $\overline{\mathbf{fwd}}(\kappa)$

- Recall the trace of $\mathtt{M}_1 \triangleq f\ (\lambda x.5)$

$$t_{\mathtt{M}_1} = \overline{f}(g, c)\ g(\mathtt{A},\ d)\ \overline{d}(5)\ c(\mathtt{true})$$

representing the interaction with the environment given by the evaluation context

$$\text{let } f = (\lambda g.g\ \mathtt{V}; \textbf{return}\ \mathtt{true}) \text{ in } []$$

- Recall the trace of $\mathtt{M}_1 \triangleq f\,(\lambda x.5)$

$$t_{\mathtt{M}_1} = \overline{f}(g, c)\ g(\mathtt{A},\ d)\ \overline{d}(5)\ c(\mathtt{true})$$

representing the interaction with the environment given by the evaluation context

$$\mathsf{let}\, f = (\lambda g.g\ \mathtt{V}; \mathbf{return}\ \mathtt{true})\ \mathsf{in}\, []$$

- Recall that the following term is equivalent to $\mathtt{M}_1$

$$\begin{array}{lll} & & \mathtt{let}\ y = \mathtt{new}\,\mathtt{E}\ \mathtt{in} \\ & & \mathbf{handle} \\ \mathtt{M}_2 & \triangleq & \quad f\,(\lambda x.y\#\mathtt{op}\,()) \\ & & \mathbf{with}\ \{\mathbf{return}\ x \mapsto \mathbf{return}\,x\} \\ & & \qquad \{y\#\mathtt{op}\ x\ \kappa \mapsto \kappa\ 5\} \end{array}$$

Now we look at how $\mathtt{M}_2$ interacts with the same environement

$$\text{let}\, f = (\lambda g.g\ \mathtt{V}; \textbf{return}\ \mathtt{true})\ \text{in}\, []$$

Now we look at how $\mathtt{M}_2$ interacts with the same environement

$$\text{let } f = (\lambda g.g\ \mathtt{V}; \textbf{return}\ \mathtt{true})\ \text{in}\ []$$

$\mathtt{M}_2$ evaluates to

$$\textbf{handle}\ f\ (\lambda x.\iota\#\mathtt{op}\ ())\ \textbf{with}\ \{\iota\#\mathtt{op}\ x\ \kappa \mapsto \kappa\ 5\}$$

then ..

Now we look at how $\mathtt{M}_2$ interacts with the same environement

$$\text{let } f = (\lambda g.g\ \mathtt{V}; \textbf{return}\ \mathtt{true})\ \text{in}\ []$$

$\mathtt{M}_2$ evaluates to

$$\textbf{handle}\ f\ (\lambda x.\iota\#\mathtt{op}\,())\ \textbf{with}\ \{\iota\#\mathtt{op}\ x\ \kappa \mapsto \kappa\ 5\}$$

then ..

$$t_{\mathtt{M}_2} = \overline{f}(g, c)$$

Now we look at how $\mathtt{M}_2$ interacts with the same environement

$$\text{let } f = (\lambda g.g\ \mathtt{V}; \textbf{return}\ \mathtt{true}) \text{ in } []$$

$\mathtt{M}_2$ evaluates to

$$\textbf{handle}\ f\ (\lambda x.\iota\#\mathtt{op}\,())\ \textbf{with}\ \{\iota\#\mathtt{op}\ x\ \kappa \mapsto \kappa\ 5\}$$

then ..

$$t_{\mathtt{M}_2} = \overline{f}(g, c)\ g(\mathtt{A}, d)$$

Now we look at how $\mathtt{M}_2$ interacts with the same environement

$$\text{let } f = (\lambda g.g\ \mathtt{V}; \textbf{return}\ \mathtt{true})\ \text{in}\ []$$

$\mathtt{M}_2$ evaluates to

$$\textbf{handle}\ f\ (\lambda x.\iota\#\mathtt{op}\,())\ \textbf{with}\ \{\iota\#\mathtt{op}\ x\ \kappa \mapsto \kappa\ 5\}$$

then ..

$$t_{\mathtt{M}_2} = \overline{f}(g, c)\ g(\mathtt{A}, d)\ \overline{\textbf{fwd}}(\kappa_d)$$

Now we look at how $\mathtt{M}_2$ interacts with the same environement

$$\text{let } f = (\lambda g.g\ \mathtt{V}; \textbf{return}\ \mathtt{true})\ \text{in}\ []$$

$\mathtt{M}_2$ evaluates to

$$\textbf{handle}\ f\ (\lambda x.\iota\#\mathtt{op}\,())\ \textbf{with}\ \{\iota\#\mathtt{op}\ x\ \kappa \mapsto \kappa\ 5\}$$

then ..

$$t_{\mathtt{M}_2} = \overline{f}(g, c)\ g(\mathtt{A}, d)\ \overline{\textbf{fwd}}(\kappa_d)\ \overline{\kappa_d}(5, c')$$

Now we look at how $\mathtt{M}_2$ interacts with the same environement

$$\text{let } f = (\lambda g. g \ \mathtt{V}; \textbf{return} \ \mathtt{true}) \text{ in } []$$

$\mathtt{M}_2$ evaluates to

$$\textbf{handle} \ f \ (\lambda x. \iota\#\mathtt{op}\,()) \ \textbf{with} \ \{\iota\#\mathtt{op} \ x \ \kappa \mapsto \kappa \ 5\}$$

then ..

$$t_{\mathtt{M}_2} = \overline{f}(g, c) \ g(\mathtt{A}, d) \ \overline{\textbf{fwd}}(\kappa_d) \ \overline{\kappa_d}(5, c') \ c'(\mathtt{true})$$

Because of this, we get

$$[\![\mathtt{M}_{\mathbf{1}}]\!]_{\mathsf{ogs}} \neq [\![\mathtt{M}_{\mathbf{2}}]\!]_{\mathsf{ogs}}$$

Because of this, we get

$$[\![\mathtt{M}_{\mathbf{1}}]\!]_{\mathsf{ogs}} \neq [\![\mathtt{M}_{\mathbf{2}}]\!]_{\mathsf{ogs}}$$

We need a coarser notion of trace equivalence in which

$$\overline{f}(g, c)\ g(\mathtt{A},\ d)\ \overline{d}(5)\ c(\mathtt{true})$$
$$\sim_{tr}$$
$$\overline{f}(g, c)\ g(\mathtt{A}, d)\ \overline{\mathbf{fwd}}(\kappa_d)\ \overline{\kappa_d}(5, c')\ c'(\mathtt{true})$$

Full-abstraction

Theorem (Soundness)

$$\simeq_{tr} \quad \subseteq \quad \simeq_{ctx}$$

Full-abstraction

Theorem (Soundness)

$$\simeq_{tr} \quad \subseteq \quad \simeq_{ctx}$$

Conjecture (Completeness)

$$\simeq_{ctx} \quad \subseteq \quad \simeq_{tr}$$

Conclusion

- Contextual equivalence is more subtle in the presence of generativity of first-class effect instances.
- Extending *OGS* model to account for observable and private effectful behaviour.
- Relaxing trace equivalence to coincide with the contextual one.

QUESTIONS?

References

[1] Gordon Plotkin and John Power. "Semantics for algebraic operations". In: *Electronic Notes in Theoretical Computer Science* 45 (2001), pp. 332–345.

[2] Andrej Bauer and Matija Pretnar. "Programming with algebraic effects and handlers". In: *Journal of Logical and Algebraic Methods in Programming* 84.1 (2015). Special Issue: The 23rd Nordic Workshop on Programming Theory (NWPT 2011) Special Issue: Domains X, International workshop on Domain Theory and applications, Swansea, 5-7 September, 2011, pp. 108–123.

[3] Dariusz Biernacki et al. "Handle with care: relational interpretation of algebraic effects and handlers". In: *Proceedings of the ACM on Programming Languages* 2.POPL (2017), pp. 1–30.

[4] Dariusz Biernacki, Sergueï Lenglet, and Piotr Polesiuk. "A complete normal-form bisimilarity for algebraic effects and handlers". In: *Formal Structures for Computation and Deduction*. 2020.

Operational Semantics

$$(\mathcal{E}[\texttt{new}\,\texttt{E}];\mathcal{V}) \mapsto (\mathcal{E}[\textbf{return}\ \iota];\mathcal{V} \uplus \{\iota\})$$

$$\begin{aligned}&(\mathcal{E}[\textbf{handle}\ (\textbf{return}\ \texttt{V})\ \textbf{with}\ \texttt{H}];\mathcal{V})\\ &\qquad \mapsto (\mathcal{E}[\texttt{M}\{x := \texttt{V}\}];\mathcal{V}) \quad \text{when } \texttt{H}^{\textbf{return}} = \{\textbf{return}\ \ x \mapsto \texttt{M}\}\end{aligned}$$

$$\begin{aligned}&(\mathcal{E}[\textbf{handle}\ \mathcal{E}'[\iota\#\texttt{op}\ \texttt{V}]\ \textbf{with}\ \texttt{H}];\mathcal{V})\\ &\quad \mapsto (\mathcal{E}[\texttt{M}\{x := \texttt{V}\}\{\kappa := \lambda y.\textbf{handle}\ \mathcal{E}'[\textbf{return}\ y]\ \textbf{with}\ \texttt{H}\}];\mathcal{V})\\ &\quad \text{when } \texttt{H}^{\texttt{op}} = \{\iota\#\texttt{op}\ x\ \kappa \mapsto \texttt{M}\}\\ &\quad \text{and } \ \iota\#\texttt{op} \notin \mathrm{hdl}(\mathcal{E}')\end{aligned}$$