Operational Game Semantics for generative algebraic effects and handlers

(work in progress)

Hamza JAAFAR, Guilhem JABER

Jan 14, 2024

Nantes Universite, LS2N, INRIA Gallinette

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

$$\mathrm{choose}(\mathtt{K}[\mathtt{t}], \mathtt{K}[\mathtt{u}]) \simeq_{\mathsf{op}} \mathtt{K}[\mathrm{choose}(\mathtt{t}, \mathtt{u})]$$

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

[1] Plotkin and Power, "Semantics for algebraic operations".

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

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

- Performing an effect: **op** `v`
- Handling an effect:

$$\begin{aligned} \mathtt{H} = \quad & \{\textbf{ret}\, x \mapsto \mathtt{u}\} && (\textit{return case}) \\ & \{\textbf{op}\, p\, \kappa \mapsto \mathtt{t}\} && (\text{op } \textit{case}) \end{aligned}$$

Operations and effect handlers, concretely

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

Example (Global state)

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

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

Generally, how to deal with multiple occurences of the same effect type $\mathbb{E}$ without forefeiting modularity?

The Eff[2] approach

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

- Performing an effect: $\iota\#\text{op}_{\mathbb{E}}\ \texttt{v}$
- Handling an effect: $\texttt{H} = \{\iota\#\text{op}_{\mathbb{E}}\, p\, \kappa \mapsto \texttt{t}\}$ ($\iota\#\text{op}_{\mathbb{E}}$ case)

[2] Bauer and Pretnar, "Programming with algebraic effects and handlers".

Programming Language

Syntax: Fine-grained call-by-value

values	$\mathtt{v}, \mathtt{w}$	$:=$	$x \mid \lambda x : \tau.\mathtt{t} \mid \iota$
Terms	$\mathtt{t}, \mathtt{u}$	$:=$	$\textbf{ret}\ \mathtt{v} \mid \mathtt{v}\ \mathtt{v} \mid \mathtt{match}\ \mathtt{v}\ \textbf{with}\ (\mathtt{P}_i \to \mathtt{u}_i)_{i \in I}$
			$\mid \mathsf{let}\, x = \mathtt{t}\, \mathsf{in}\, \mathtt{u}$
			$\mid \mathtt{v}\textbf{\#op}\ \mathtt{w} \mid \{\mathtt{t}\}\, \textbf{with}\, \mathtt{H}$
Handlers	$\mathtt{H}$	$:=$	$\{\textbf{ret}\, x \mapsto \mathtt{t}\} \mid \{\mathtt{v}\textbf{\#op}\, x\, \kappa \mapsto \mathtt{t}\} \uplus \mathtt{H}$
ECxts	$\mathtt{K}$	$:=$	$\bullet \mid \mathsf{let}\, x = \mathtt{K}\, \mathsf{in}\, \mathtt{t} \mid \{\mathtt{K}\}\, \textbf{with}\, \mathtt{H}$

Dynamic generation of effects

New construct

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

New construct: $\mathtt{t}, \mathtt{u} \quad := \quad \cdots \mid \textbf{new}\, \mathbb{E}$

Operational Semantics: $(\textbf{new}\, \mathbb{E}; \mathcal{V}) \mapsto_{\mathsf{op}} (\textbf{ret}\ \iota; \mathcal{V} \uplus \{\iota\})$

Disclosure and contextual equivalence

Consider the following variation of an example from[3]

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

$$\simeq_{ctx}$$

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

[3] Biernacki, Piróg, et al., "Handle with care: relational interpretation of algebraic effects and handlers".

Disclosure and contextual equivalence (cont.)

Now consider a variation of the previous example:

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

$$\not\simeq_{ctx}$$

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

Operational game semantics model

Existing fully-abstract models for effect handlers

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

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

[4] Biernacki, Lenglet, and Polesiuk, "A complete normal-form bisimilarity for algebraic effects and handlers".

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:

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

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

    $$\bar{c}(\mathtt{A}) \qquad | \qquad c(\mathtt{A})$$

Examples

Let's consider the trace of

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

representing the interaction with the environment given by the evaluation context

$$\text{let } f = (\lambda g.g\ \mathtt{v}; \textbf{ret}\ \mathtt{tt}) \text{ in } []$$

yielding the trace

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

Operational Game Semantics (cont.)

Normal Forms:

$$\mathtt{Nf} = \mathtt{K}[f\,V] \mid \mathbf{ret}\ V$$

- $\mathtt{K}[f\,V]$ calls for a P-question of the shape $\bar{f}(\mathtt{A}, c)$
- $\mathbf{ret}\ V$ calls for an answer of the shape $\bar{c}(\mathtt{A})$

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

OGS model for algebraic effects and handlers

Algebraic effects introduce new normal forms:

$$\texttt{Nf} = \cdots \mid \texttt{K}[\iota \# \mathit{op}\, V] \qquad \text{when } \iota\#\mathbf{op} \notin \mathbf{hdl}(\texttt{K})$$

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

$$\iota\#\mathsf{op}\ \mathtt{v}$$

- 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:

$$\mathtt{Nf} = \cdots \mid \mathtt{K}[\iota\#\mathit{op}\, V] \qquad \text{when } \iota\#\mathbf{op} \notin \mathbf{hdl}(\mathtt{K})$$

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

- Recall the trace of $\mathtt{t}_1 := f(\lambda x.5)$

$$t_{\mathtt{t}_1} = \bar{f}(g, c)\ g(\mathtt{A},\ d)\ \bar{d}(5)\ c(\mathtt{tt})$$

representing the interaction with the environment given by the evaluation context

$$\text{let } f = (\lambda g.g\ \mathtt{v}; \textbf{ret}\ \mathtt{tt})\ \text{in}\ []$$

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

$$\mathtt{t}_2 \quad := \quad \begin{array}{l} \mathtt{let}\ y = \textbf{new}\ \mathbb{E}\ \ \mathtt{in} \\ \textbf{handle} \\ \quad f(\lambda x. y\#\textbf{op}\ \langle\rangle) \\ \textbf{with}\ \{\textbf{ret}\ x \mapsto \textbf{ret}\ x\} \\ \qquad \{y\#\textbf{op}\ x\ \kappa \mapsto \kappa\ 5\} \end{array}$$

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

$$\text{let } f = (\lambda g.g\ \mathtt{v}; \textbf{ret}\ \mathtt{tt}) \text{ in } []$$

$\mathtt{t}_2$ evaluates to

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

then ..

$$t_{\mathtt{t}_2} = \bar{f}(g, c)\ g(\mathtt{A}, d)\ \overline{\textbf{fwd}}(\kappa_d)\ \overline{\kappa_d}(5, c')\ c'(\mathtt{tt})$$

Because of this, we get

$$[\![\mathtt{t_1}]\!]_{\mathsf{ogs}} \neq [\![\mathtt{t_2}]\!]_{\mathsf{ogs}}$$

We need a coarser notion of trace equivalence in which

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

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?

- Plotkin, Gordon and John Power. **"Semantics for algebraic operations".** In: *Electronic Notes in Theoretical Computer Science* 45 (2001), pp. 332–345.
- Bauer, Andrej 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.
- Biernacki, Dariusz, Maciej Piróg, 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.

Biernacki, Dariusz, 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

$$(\mathtt{K}[\textbf{new}\ \mathbb{E}]; \mathcal{V}) \mapsto_{\textsf{op}} (\mathtt{K}[\textbf{ret}\ \iota]; \mathcal{V} \uplus \{\iota\})$$

$$\begin{aligned}&(\mathtt{K}[\{(\textbf{ret}\ \mathtt{v})\}\,\textbf{with}\,\mathtt{H}]; \mathcal{V})\\ &\qquad \mapsto_{\textsf{op}} (\mathtt{K}[\mathtt{t}\{x := \mathtt{v}\}]; \mathcal{V}) \quad \text{when } \mathtt{H}^{\textbf{ret}} = \{\textbf{ret}\ x \mapsto \mathtt{t}\}\end{aligned}$$

$$\begin{aligned}&(\mathtt{K}[\{\mathtt{K}'[\iota\#\textbf{op}\ \mathtt{v}]\}\,\textbf{with}\,\mathtt{H}]; \mathcal{V})\\ &\qquad \mapsto_{\textsf{op}} (\mathtt{K}[\mathtt{t}\{x := \mathtt{v}\}\{\kappa := \lambda y.\{\mathtt{K}'[\textbf{ret}\ y]\}\,\textbf{with}\,\mathtt{H}\}]; \mathcal{V})\\ &\quad \text{when } \mathtt{H}^{\textbf{op}} = \{\iota\#\textbf{op}\,x\,\kappa \mapsto \mathtt{t}\}\\ &\quad \text{and } \ \iota\#\textbf{op} \notin \textbf{hdl}(\mathtt{K}')\end{aligned}$$