Algebraic Effect Handlers for WASM

ANDREAS ROSSBERG ET AL., Dfinity

Algebraic effect handlers are a powerful abstraction mechanism that can express many complex control-flow mechanisms.

updates:
v1, 2018-04-03: Initial version.

1. INTRODUCTION

Algebraic effects [10] and their extension with handlers [11, 12], are a novel way to describe many control-flow mechanisms in programming languages. In general any free monad can be expressed as an effect handler and they have been used to describe complex control structures such as iterators, async-await, concurrency, parsers, state, exceptions, etc. (without needing to extend the compiler or language) [3–6, 14].

Recently, there are various implementations of algebraic effects, either embedded in other languages like Haskell [5, 14], Scala [2], or C [7], or built into a language, like Eff [1], Links [4], Frank [9], Koka [8], and Multi-core OCaml [3, 13].

2. FORMALIZATION

Syntax

reference types.

rt ::= ...
| $(exn\ \tau^*)$
| $(cont\ \varphi)$

q ::= exn | eff

instructions.

e ::= ...
| throw x
| rethrow
| resume
| abort
| try$_q$ φ e^* catch e^* end
| handle$_q$ φ x e^* *else* e^* end

administrative instructions.

e ::= ...
| catch$_n^q$ e^* e^* end
| throw a
| swallow a
| exn$_n$ a v^* $a^?$
| cont$_n$ a

exception definitions.

e ::=
| ex^* $exception_q$ φ
| ex^* $exception_q$ φ im

2.1. Typing

contexts.

$C \;::=\; \ldots,\ exn\,(q\,\varphi)^*$

$$\frac{C_{exn}(x) \;=\; \varphi}{C \vdash \mathsf{throw}\ x \;:\; \varphi}$$

$$\frac{}{C \vdash \mathsf{rethrow} \;:\; (exn\,\tau^*) \to \tau^*}$$

$$\frac{}{C \vdash \mathsf{resume} \;:\; \tau_1^*\ (cont\,\tau_1^* \to \tau_2^*) \to \tau_2^*}$$

$$\frac{}{C \vdash \mathsf{abort} \;:\; (cont\,\tau_1^* \to \tau_2^*) \to \tau_2^*}$$

$$\frac{\begin{array}{l} \varphi \;=\; \tau_1^* \to \tau_2^* \\ C^q, label\,\tau_2^* \vdash e_1^* \;:\; \varphi \\ C^q, label\,\tau_2^* \vdash e_2^* \;:\; (exn\,\tau_2^*) \to \tau_2^* \end{array}}{\begin{array}{l} C \vdash \mathsf{try}_q\ \varphi\ e_1^*\ \mathsf{catch}\ e_2^*\ \mathsf{end} \;:\; \varphi \\ \text{where} \\ \quad C^{exn} \;=\; C \\ \quad C^{eff} \;=\; C\ \text{with}\ label \;=\; . \end{array}}$$

$$\frac{\begin{array}{l} \varphi \;=\; \tau_1^* \to \tau_2^* \\ C_{exn}(x) \;=\; q\,(\tau_3^* \to \tau_4^*) \\ q \;=\; exn \;\wedge\; \tau'? \;=\; . \;\vee\; q \;=\; eff \;\wedge \\ \tau'? \;=\; (cont\,\tau_2^*\,\tau^*)\ C, label\,\tau_2^* \vdash e_1^* \;:\; \tau_1^*\,\tau_3^* \\ \tau'? \to \tau_2^*\ C, label\,\tau_2^* \vdash e_2^* \;:\; \tau_1^*\,(exn\,\tau^*) \to \tau_2^* \end{array}}{C \vdash \mathsf{handle}_q\ \varphi\ x\ e_1^*\ else\ e_2^*\ \mathsf{end} \;:\; \tau_1^*\,(exn\,\tau^*) \to \tau_2^*}$$

$$\frac{\begin{array}{l} S;\ C^q, label\,\tau_2^* \vdash e_1^* \;:\; \tau_1^* \to \tau_2^* \\ S;\ C^q, label\,\tau_2^* \vdash e_2^* \;:\; (exn\,\tau_2^*) \to \tau_2^* \end{array}}{S;\ C \vdash \mathsf{catch}_n^q e_2^*\ e_1^*\ \mathsf{end} \;:\; \tau_1^* \to \tau_2^*}$$

$$\frac{S_{exn}(a) \;=\; q\,\varphi}{S;\ C \vdash \mathsf{throw}\ a \;:\; \varphi}$$

$$\frac{S_{exn}(a) \;=\; q\,\varphi}{S;\ C \vdash \mathsf{swallow}\ a \;:\; (exn\,\tau^*) \to .}$$

$$\frac{\begin{array}{l} S_{exn}(a) \;=\; exn\,(\tau_1^* \to .) \\ (C \vdash v \;:\; \tau_1)^* \end{array}}{S;\ C \vdash exn_n\ a\ v^* \;:\; exn\,\tau^*}$$

$$\frac{\begin{array}{l} S_{exn}(a_1) \;=\; eff\,(\tau_1^* \to \tau_2^*) \\ (C \vdash v \;:\; \tau_1)^* \\ S;\ . \vdash S_{cont}(a_2) \;:\; (. \to \tau_2^*) \;\Rightarrow\; (. \to \tau^*) \end{array}}{S;\ C \vdash exn_n\ a_1\ v^*\ a_2 \;:\; exn\,\tau^*}$$

$$\frac{\mathsf{S}_{cont}(a) \;=\; \mathsf{E}_{\mathit{eff}} \qquad \mathsf{S};\ .\ \vdash\ \mathsf{E}_{\mathit{eff}}\ :\ (. \to \tau_1^*)\ \Rightarrow\ (. \to \tau_2^*)}{\mathsf{S};\ C\ \vdash\ \mathit{cont}_n\ a\ :\ \mathit{cont}\ (\tau_1^* \to \tau_2^*)}$$

2.2. Reduction

module instance.

$M\ ::=\ \ldots,\ \mathit{exn}\ a^*$

store.

$\mathsf{S}\ ::=\ \ldots,\ \mathit{exn}\ (q\ \varphi)^*,\ \mathit{cont}\ (\mathsf{E}^?)^*$

lookup.

$\mathsf{F}_{exn}(x)\ :=\ (\mathsf{F}_{mod})_{exn}(x)$

branch contexts.

$B^0\ ::=\ v^*\ \_\ e^*$

$B^(i+1)\ ::=\ \mathit{label}_n e^*\ B^i\ \mathsf{end}\ |\ \mathsf{catch}_m^q\ e^*\ B^(i+1)\ \mathsf{end}$

throw contexts.

$\mathsf{E}_q\ ::=\ v^*\ \_\ e^*\ |\ \mathit{label}_n e^*\ \mathsf{E}_q\ \mathsf{end}\ |\ \mathsf{catch}_m^{(\ q)}\ e^*\ \mathsf{E}_q\ \mathsf{end}\ |\ \mathit{frame}_n \mathsf{F}\ \mathsf{E}_q\ \mathsf{end}$

$\mathsf{F};\ \mathsf{throw}\ x\ \longrightarrow\ \mathsf{F};\ \mathsf{throw}\ a$
  $\mathsf{F}_{exn}(x)\ =\ a$

$v^n\ (\mathsf{try}_q\ \varphi\ e_1^*\ \mathsf{catch}\ e_2^*\ \mathsf{end})\ \longrightarrow\ \mathsf{catch}_m^q e_2^*\ (\mathit{label}_m.\ v^n\ e_1^*\ \mathsf{end})\ \mathsf{end}$
  $\varphi\ =\ \tau_1^n \to \tau_2^m$

$\mathsf{catch}_m^q e^*\ v^*\ \mathsf{end}\ \longrightarrow\ v^*$

$\mathsf{S};\ \mathsf{F};\ \mathsf{catch}_m^{exn} e^*\ \mathsf{E}_{exn}[v^n\ (\mathsf{throw}\ a_1)]\ \mathsf{end}\ \longrightarrow\ \mathsf{S}';\ \mathsf{F};\ \mathit{label}_m.\ (\mathit{exn}_m\ a_1\ v^n)\ e^*\ \mathsf{end}$
  $\mathsf{S}_{exn}(a_1)\ =\ \mathit{exn}\ (\tau_1^n \to \tau_2^m)$

$\mathsf{S};\ \mathsf{F};\ \mathsf{catch}_m^{\mathit{eff}}\ e^*\ \mathsf{E}_{\mathit{eff}}[v^n\ (\mathsf{throw}\ a_1)]\ \mathsf{end}\ \longrightarrow\ \mathsf{S}';\ \mathsf{F};\ \mathit{label}_m.\ (\mathit{exn}_m\ a_1\ v^n\ a_2)\ e^*\ \mathsf{end}$
  $\mathsf{S}_{exn}(a_1)\ =\ \mathit{eff}\ (\tau_1^n \to \tau_2^m)$
  $a_2\ =\ |\mathsf{S}_{cont}|$
  $\mathsf{S}'\ =\ \mathsf{S}\ \mathit{with}\ \mathit{cont} +=\ \mathsf{E}'$
  $\mathsf{E}'\ =\ \mathsf{catch}_m^{\mathit{eff}}\ e^*\ \mathsf{E}_{\mathit{eff}}\ \mathsf{end}$

$(\mathit{exn}_m\ a_1\ v^n\ a_2^?)\ \mathsf{rethrow}\ \longrightarrow\ v^n\ (\mathsf{throw}\ a_1)\ (\ (\mathit{cont}_m\ a_2)\ \mathsf{resume}\ )^?$

$\mathsf{F};\ v_1^n\ (\mathit{exn}_m\ a_1\ v^*\ a_2^?)\ \mathsf{handle}_q\ \varphi\ x\ e_1^*\ \mathit{else}\ e_2^*\ \mathsf{end}\ \longrightarrow\ \mathsf{F};\ \mathit{label}_k.\ v_1^n\ v^*\ (\mathit{cont}_m\ a_2)^?\ e_1^*\ \mathsf{end}$
  $\mathsf{F}_{exn}(x)\ =\ a_1$
  $\varphi\ =\ \tau_1^n \to \tau_2^k$

$\mathsf{F};\ v_1^n\ (\mathit{exn}_m\ a_1\ v^*\ a_2^?)\ \mathsf{handle}_q\ \varphi\ x\ e_1^*\ \mathit{else}\ e_2^*\ \mathsf{end}\ \longrightarrow\ \mathsf{F};\ \mathit{label}_k.\ v_1^n\ (\mathit{exn}_m\ a_1\ v^*\ a_2^?)\ e_2^*\ \mathsf{end}$
  $\mathsf{F}_{exn}(x)\ = /=\ a_1$
  $\varphi\ =\ \tau_1^n \to \tau_2^k$

$\mathsf{S};\ v^n\ (\mathit{cont}_n\ a)\ \mathsf{resume}\ \longrightarrow\ \mathsf{S}';\ \mathsf{E}_{\mathit{eff}}[v^n]$
  $\mathsf{S}_{cont}(a)\ =\ \mathsf{E}_{\mathit{eff}}$
  $\mathsf{S}'\ =\ \mathsf{S}\ \mathit{with}\ \mathit{cont}(a)\ =\ .$

$\mathsf{S};\ v^n\ (\mathit{cont}_n\ a)\ \mathsf{resume}\ \longrightarrow\ \mathsf{S};\ \mathit{trap}$
  $\mathsf{S}_{cont}(a)\ =\ .$

$$\mathsf{S};\ (cont_n\ a)\ \mathsf{abort}\ \longrightarrow\ \mathsf{S}';\ \mathsf{catch}_n^{exn}\mathsf{swallow}\ a\ \mathsf{E}_{eff}[(\mathsf{throw}\ a')]\ \mathsf{end}$$

$$\mathsf{S}_{cont}(a)\ =\ \mathsf{E}_{eff}$$

$$a'\ =\ |\mathsf{S}_{exn}|$$

$$\mathsf{S}'\ =\ \mathsf{S}\ \mathsf{with}\ exn\ +=\ exn\ (.\ \rightarrow .)\ with\ cont(a)\ =\ .$$

$$\mathsf{S};\ (cont_n\ a)\ \mathsf{abort}\ \longrightarrow\ \mathsf{S};\ trap$$

$$\mathsf{S}_{cont}(a)\ =\ .$$

$$(exn_n\ a_1\ v^*\ a_2^?)\ (\mathsf{swallow}\ a_1)\ \longrightarrow\ .$$

$$(exn_n\ a_1\ v^*\ a_2^?)\ (\mathsf{swallow}\ a_3)\ \longrightarrow\ (exn_n\ a_1\ v^*\ a_2^?)\ \mathsf{rethrow}$$

$$a_1\ =/=\ a_3$$

3. CONCLUSION

REFERENCES

[1] Andrej Bauer, and Matija Pretnar. "Programming with Algebraic Effects and Handlers." *J. Log. Algebr. Meth. Program.* 84 (1): 108–123. 2015. doi:10.1016/j.jlamp.2014.02.001.

[2] Jonathan Immanuel Brachthäuser, and Philipp Schuster. "Effekt: Extensible Algebraic Effects in Scala." In *Scala'17*. Vancouver, CA. Oct. 2017.

[3] Stephen Dolan, Spiros Eliopoulos, Daniel Hillerström, Anil Madhavapeddy, KC Sivaramakrishnan, and Leo White. "Concurrent System Programming with Effect Handlers." In *Proceedings of the Symposium on Trends in Functional Programming*. TFP'17. May 2017.

[4] Daniel Hillerström, and Sam Lindley. "Liberating Effects with Rows and Handlers." In *Proceedings of the 1st International Workshop on Type-Driven Development*, 15–27. TyDe 2016. Nara, Japan. 2016. doi:10.1145/2976022.2976033.

[5] Ohad Kammar, Sam Lindley, and Nicolas Oury. "Handlers in Action." In *Proceedings of the 18th ACM SIGPLAN International Conference on Functional Programming*, 145–158. ICFP '13. ACM, New York, NY, USA. 2013. doi:10.1145/2500365.2500590.

[6] Daan Leijen. "Structured Asynchrony with Algebraic Effects." In *Proceedings of the 2nd ACM SIGPLAN International Workshop on Type-Driven Development*, 16–29. TyDe 2017. Oxford, UK. 2017. doi:10.1145/3122975.3122977.

[7] Daan Leijen. "Implementing Algebraic Effects in C." In *Programming Languages and Systems*, edited by Bor-Yuh Evan Chang, 339–363. Springer International Publishing. 2017.

[8] Daan Leijen. "Type Directed Compilation of Row-Typed Algebraic Effects." In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL'17)*, 486–499. Paris, France. Jan. 2017. doi:10.1145/3009837.3009872.

[9] Sam Lindley, Connor McBride, and Craig McLaughlin. "Do Be Do Be Do." In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL'17)*, 500–514. Paris, France. Jan. 2017. doi:10.1145/3009837.3009897.

[10] Gordon D. Plotkin, and John Power. "Algebraic Operations and Generic Effects." *Applied Categorical Structures* 11 (1): 69–94. 2003. doi:10.1023/A:1023064908962.

[11] Gordon D. Plotkin, and Matija Pretnar. "Handlers of Algebraic Effects." In *18th European Symposium on Programming Languages and Systems*, 80–94. ESOP'09. York, UK. Mar. 2009. doi:10.1007/978-3-642-00590-9_7.

[12] Gordon D. Plotkin, and Matija Pretnar. "Handling Algebraic Effects." In *Logical Methods in Computer Science*, volume 9. 4. 2013. doi:10.2168/LMCS-9(4:23)2013.

[13] Leo White. "Effect Types for OCaml." Sep. 2016. `https://github.com/lpw25/ocaml-typed-effects`.

[14] Nicolas Wu, Tom Schrijvers, and Ralf Hinze. "Effect Handlers in Scope." In *Proceedings of the 2014 ACM SIGPLAN Symposium on Haskell*, 1–12. Haskell '14. Göthenburg, Sweden. 2014. doi:10.1145/2633357.2633358.