On the expressive power of user-defined effects: effect handlers, monadic reflection, and delimited control

Yannick Forster, **Ohad Kammar**,
Sam Lindley, and Matija Pretnar

22nd ACM SIGPLAN
International Conference on Functional Programming
4 September 2017
Oxford, United Kingdon, EU

User-defined effects

User-defined state

$$\begin{aligned} toggle = \{\, & x \leftarrow get!; \\ & y \leftarrow not!\ x; \\ & put!\ y; \\ & x\} \end{aligned}$$

Direct-style

$$\begin{aligned} get &= \{\quad \lambda s.(\ s, s\ )\} \\ put &= \{\lambda s'.\lambda\_.(\,(), s')\} \\ runState &= \lambda c.\lambda s.c!\ s \\ toggle &= \{\lambda s.\ (x, s) \leftarrow get!\ s; \\ &\qquad\quad y \leftarrow not!\ x; \\ &\qquad\quad (\_, s) \leftarrow put!\ y\ s; \\ &\qquad\quad (x, s)\} \end{aligned}$$

State-passing

User-defined effects

User-defined state

$$
\begin{aligned}
get &= \{\ \lambda s.(\ s, s\ )\} \\
put &= \{\lambda s'.\lambda\_.(\,(), s')\} \\
runState &= \lambda c.\lambda s.c!\ s
\end{aligned}
$$

$$
\begin{aligned}
toggle = \{\, & x \leftarrow get!; \\
& y \leftarrow not!\ x; \\
& put!\ y; \\
& x\}
\end{aligned}
\qquad
\begin{aligned}
toggle = \{\lambda s.\, & (x, s) \leftarrow get!\ s; \\
& y \leftarrow not!\ x; \\
& (\_, s) \leftarrow put!\ y\ s; \\
& (x, s)\}
\end{aligned}
$$

Direct-style — State-passing

Macro-expressibility

A **macro translation**:

- Local
- Preserves core constructs

Goal

Relative expressiveness in language design

Compare and contrast:

- Algebraic effects and handlers
- Monads
- Delimited control

Goal

Relative expressiveness in language design

Compare and contrast:

- Algebraic effects and handlers
- Monads
- Delimited control

Small print

- Large design space:
  deep handlers, shallow handlers, parameterised monads, graded monads, shift vs. shift0, answer-type modification
- Inexpressivity is brittle:
  adding inductive or polymorphic types invalidates our proofs

Goal

Relative expressiveness in language design

Compare and contrast:

- Algebraic effects and handlers
- Monads
- Delimited control

Small print

- Large design space
- Inexpressivity is brittle

Takeaway message

Expressibility must be stated as formal translations between calculi.

Contribution

status
established

$\longrightarrow$

CBPV

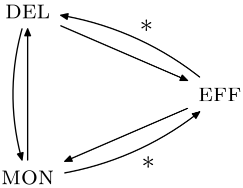
Contribution

$$\begin{tikzcd} \mathrm{DEL} & & \\ & \mathrm{CBPV} \arrow[ul, hook] \arrow[r, hook] \arrow[dl, hook] & \mathrm{EFF} \\ \mathrm{MON} & & \end{tikzcd}$$

$$\xrightarrow{\frac{\text{status}}{\text{established}}}$$

- Syntax, operational semantics
- Formalisation in Abella

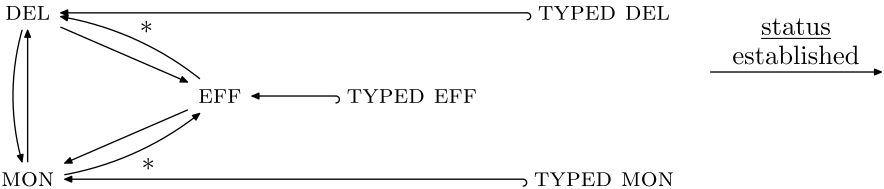
Contribution



$\xrightarrow[\text{established}]{\text{status}}$

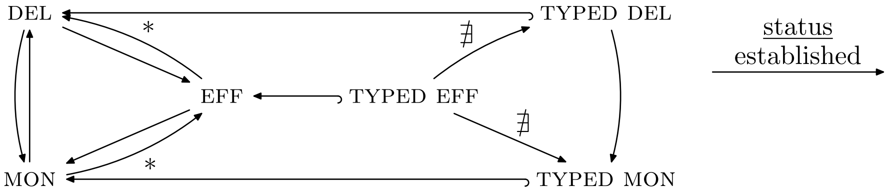
- Syntax, operational semantics
- Macro translations
- Formalisation in Abella

Contribution



- Syntax, operational semantics
- Macro translations
- Denotational semantics for EFF, MON
- Meta-theory:
  - Type safety
  - Adequacy and soundness for EFF, MON
- Formalisation in Abella

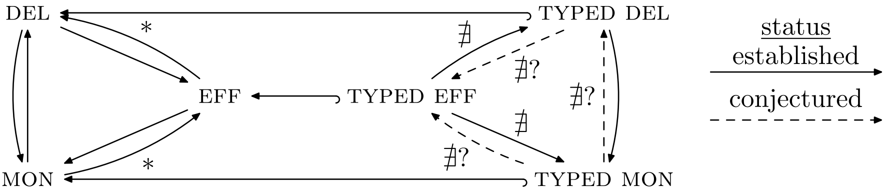
Contribution



- Syntax, operational semantics
- Macro translations
- Denotational semantics for EFF, MON
- Meta-theory:
  - Type safety
  - Adequacy and soundness for EFF, MON

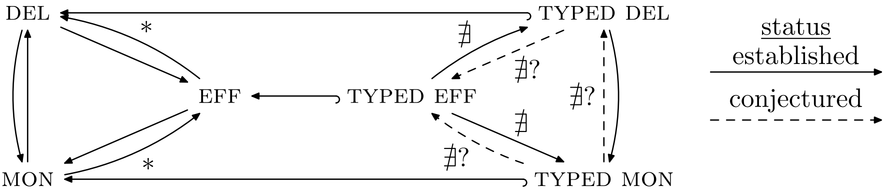
- Formalisation in Abella
- Typeability preservation proofs
- Inexpressibility proofs

Contribution



- Syntax, operational semantics
- Macro translations
- Denotational semantics for EFF, MON
- Meta-theory:
  - Type safety
  - Adequacy and soundness for EFF, MON
- Formalisation in Abella
- Typeability preservation proofs
- Inexpressibility proofs

Contribution (this talk)



- Syntax, operational semantics
- Macro translations
- Denotational semantics for EFF, MON
- Meta-theory:
  - Type safety
  - Adequacy and soundness for EFF, MON
- Formalisation in Abella
- Typeability preservation proofs
- Inexpressibility proofs

Effect handlers

User-defined state

$$
\begin{aligned}
\mathit{toggle} &= \{x \leftarrow \mathrm{get}\ ();\\
&\quad\ \ y \leftarrow \mathit{not}!\ x;\\
&\quad\ \ \mathrm{put}\ y;\\
&\quad\ \ x\}\\
H_{ST} &= \{\mathbf{return}\ \ x \mapsto \lambda s.\mathbf{return}\ x\\
&\quad\ \ \mathrm{get}\ \_\ \ k \quad \mapsto \lambda s.k!\ s\ \ s\\
&\quad\ \ \mathrm{put}\ s'\ k \quad \mapsto \lambda\_.k!\ ()\ s'\}\\
\mathit{runState} &= \{\lambda c.\mathbf{handle}\ c!\ \mathbf{with}\ H_{ST}\}
\end{aligned}
$$

Effect handlers

User-defined state

$$
\begin{aligned}
toggle &= \{x \leftarrow \mathrm{get}\ ();\\
&\qquad y \leftarrow not!\ x;\\
&\qquad \mathrm{put}\ y;\\
&\qquad x\}\\
H_{ST} &= \{\mathbf{return}\ \ x \mapsto \lambda s.\mathbf{return}\ x\\
&\qquad \mathrm{get}\ \_\ \ k \ \ \mapsto \lambda s.k!\ s\ \ s\\
&\qquad \mathrm{put}\ s'\ k \ \ \mapsto \lambda\_.k!\ ()\ s'\}\\
runState &= \{\lambda c.\mathbf{handle}\ c!\ \mathbf{with}\ H_{ST}\}
\end{aligned}
$$

$$runState!\ toggle\ \texttt{True} \leadsto^* (\mathbf{handle}\ \texttt{True}\ \mathbf{with}\ H_{ST})\ \texttt{False} \leadsto^* \texttt{True}$$

Effect handlers

User-defined state

$$
\begin{array}{ll}
\mathit{toggle} = \{x \leftarrow \mathrm{get}\ ();\ y \leftarrow \mathit{not}!\ x;\ \mathrm{put}\ y;\ x\} & : U_{\mathit{State}} F\mathbf{bit}
\end{array}
$$

$$\mathit{State} = \{\mathrm{get} : 1 \to \mathbf{bit}, \mathrm{put} : \mathbf{bit} \to 1\} : \mathbf{Eff}$$

$$
\begin{aligned}
H_{ST} = \{&\mathbf{return}\ x \mapsto \lambda s.\mathbf{return}\ x \\
&\mathrm{get}\ \_\ k \mapsto \lambda s.k!\ s\ \ s \\
&\mathrm{put}\ s'\ k \mapsto \lambda \_.k!\ ()\ s'\} \quad : \mathbf{bit}\ {}^{\mathit{State}}\!\!\Rightarrow^{\emptyset} \mathbf{bit} \to F\mathbf{bit}
\end{aligned}
$$

$$\mathit{runState} = \{\lambda c.\mathbf{handle}\ c!\ \mathbf{with}\ H_{ST}\} : U_{\emptyset}((U_{\mathit{State}} F\mathbf{bit}) \to \mathbf{bit} \to F\mathbf{bit})$$

$$\mathit{runState}!\ \mathit{toggle}\ \mathtt{True} \rightsquigarrow^{*} (\mathbf{handle}\ \mathtt{True}\ \mathbf{with}\ H_{ST})\ \mathtt{False} \rightsquigarrow^{*} \mathtt{True}$$

Monadic reflection

User-defined state

$$
\begin{aligned}
toggle &= \{x \leftarrow get!;\\
&\qquad y \leftarrow not!\ x;\\
&\qquad put!\ y;\\
&\qquad x\}\\
get &= \{\quad \mu(\lambda s.(s\ \ , s))\}\\
put &= \{\lambda s'.\mu(\lambda\_.((), s'))\}
\end{aligned}
$$

$$
\begin{aligned}
runState &= \{\lambda c.\,[c!]^{\,T_{State}}\}\\
T_{State} &= \mathbf{where}\,\{\\
&\qquad \mathbf{return}\ x = \lambda s.(x, s);\\
&\qquad f \gg\!= k \quad = \lambda s.(x, s') \leftarrow f\ s;\\
&\qquad\qquad\qquad k!\ x\ s'\}
\end{aligned}
$$

User-defined state

$$
\begin{aligned}
toggle \quad &= \{x \leftarrow get!;\\
&\quad\ y \leftarrow not!\ x;\\
&\quad\ put!\ y;\\
&\quad\ x\}\\
get \quad &= \{\quad \mu(\lambda s.(s\ \ , s))\}\\
put \quad &= \{\lambda s'.\mu(\lambda\_.((), s'))\}
\end{aligned}
$$

$$
\begin{aligned}
runState &= \{\lambda c.\,[c!]^{T_{State}}\}\\
T_{State} &= \textbf{where}\,\{\\
&\qquad \textbf{return}\ x = \lambda s.\,(x, s)\,;\\
&\qquad f \gg\!= k \quad = \lambda s.(x, s') \leftarrow f\ s;\\
&\qquad\qquad\qquad\qquad k!\ x\ s'\}
\end{aligned}
$$

$$runState!\ toggle\ \texttt{True} \rightsquigarrow^{\star} \textbf{return}\ \ (\texttt{True}, \texttt{False})$$

Monadic reflection

User-defined state

$$
\begin{aligned}
toggle \quad &= \{x \leftarrow get!;\\
&\quad\ y \leftarrow not!\ x;\\
&\quad\ put!\ y;\\
&\quad\ x\} && : U_{State} F\mathbf{bit}\\
get \quad &= \{\quad\ \mu(\lambda s.(s\ \ , s))\} && : U_{State} F\mathbf{bit}\\
put \quad &= \{\lambda s'.\mu(\lambda\_.((), s'))\} && : U_{State}(\mathbf{bit} \to F1)
\end{aligned}
$$

$$
runState = \{\lambda c.\,[c!]^{T_{State}}\} \qquad : U_{\emptyset}((U_{State}F\mathbf{bit}) \to \mathbf{bit} \to F(\mathbf{bit} \times \mathbf{bit}))
$$

$$
\begin{aligned}
State = \emptyset \prec\ &\mathbf{instance\ monad}\,(\alpha.\mathbf{bit} \to F(\alpha \times \mathbf{bit}))\\
&\mathbf{where}\,\{\\
&\quad \mathbf{return}\ x = \lambda s.(x, s);\\
&\quad f \gg\!= k \quad = \lambda s.(x, s') \leftarrow f\ \ s;\\
&\qquad\qquad\qquad\quad k!\ x\ s'\} : \mathbf{Eff}
\end{aligned}
$$

$$
runState!\ toggle\ \mathtt{True} \leadsto^{\star} \mathbf{return}\ \ (\mathtt{True}, \mathtt{False})
$$

Translation: MON→EFF

$$\underline{\mu(N)} := \text{reflect } \{\underline{N}\}$$
$$\underline{[M]^T} := \textbf{handle } \underline{M} \textbf{ with } \underline{T}$$
$$\underline{T} := \{\textbf{return } x \mapsto \underline{N_u}$$
$$\text{reflect } y\, f \mapsto \underline{N_b}\}$$

Theorem (Correctness)

EFF *simulates* MON:
$M \rightsquigarrow N \implies \underline{M} \rightsquigarrow^+ \underline{N}$

This translation does not preserve typability:

$$[\, b \leftarrow \mu(\{\lambda(b, f).b\});$$
$$f \leftarrow \mu(\{\lambda(b, f).f\});$$
$$f!\ b]^{T_{\text{Reader}}}$$
$$(\mathbf{inj}_{\text{true}}\, (), \{\lambda b.\mathbf{return}\ b\})$$

- Reflection at different type
- Remedy: effects with polymorphic arities

$$\text{Reader} = \emptyset \prec \textbf{instance monad}\,(\alpha.\mathbf{bit} \times U_\emptyset\,(\mathbf{bit} \to F\,\mathbf{bit}) \to F\alpha)$$
$$\textbf{where}\,\{\textbf{return}\ x = \lambda e.\textbf{return}\ x;$$
$$m \gg\!= f = \lambda e.x \leftarrow m!\ e;\ f!\ x\ e\}$$

Delimited control

$$\begin{aligned}
\mathit{toggle} &= \{x \leftarrow \mathit{get}!;\\
&\phantom{= \{}y \leftarrow \mathit{not}!\ x;\\
&\phantom{= \{}\mathit{put}!\ y;\\
&\phantom{= \{}x\}\\
\mathit{get} &= \{\quad\ \mathbf{S_0} k.\lambda s.k!\ s\ s\}\\
\mathit{put} &= \{\lambda s'.\mathbf{S_0} k.\lambda_.k!\ ()\ s'\}\\
\mathit{runState} &= \{\lambda c.\,\langle c! | x.\lambda s.x\rangle\}
\end{aligned}$$

(shift-zero and dollar without answer-type modification)

Delimited control

$$
\begin{aligned}
toggle &= \{x \leftarrow get!;\\
&\quad\; y \leftarrow not!\ x;\\
&\quad\; put!\ y;\\
&\quad\; x\}\\
get &= \{\quad\;\; \mathbf{S_0} k.\lambda s.k!\ s\ s\}\\
put &= \{\lambda s'.\mathbf{S_0} k.\lambda\_.k!\ ()\ s'\}\\
runState &= \{\lambda c.\ \langle c! | x.\lambda s.x\rangle\}
\end{aligned}
$$

$$runState!\ toggle\ \mathtt{True} \leadsto^* \langle \mathtt{True} | x.\lambda s.x\rangle\ \mathtt{False} \leadsto^* \textbf{return}\ \ \mathtt{True}$$

(shift-zero and dollar without answer-type modification)

Delimited control

$$
\begin{aligned}
toggle &= \{x \leftarrow get!; & State = \emptyset, \mathbf{bit} \to F\mathbf{bit} : \mathbf{Eff} \\
&\quad\; y \leftarrow not!\ x; & \\
&\quad\; put!\ y; & \\
&\quad\; x\} & : U_{State} F\mathbf{bit} \\
get &= \{\quad \mathbf{S_0} k.\lambda s.k!\ s\ s\} & : U_{State} F\mathbf{bit} \\
put &= \{\lambda s'.\mathbf{S_0} k.\lambda \_.k!\ ()\ s'\} & : U_{State}(\mathbf{bit} \to F1) \\
runState &= \{\lambda c.\, \langle c! | x.\lambda s.x\rangle\} & : U_{\emptyset}((U_{State} F\mathbf{bit}) \to \mathbf{bit} \to F\mathbf{bit})
\end{aligned}
$$

$$
runState!\ toggle\ \texttt{True} \leadsto^* \langle \texttt{True} | x.\lambda s.x\rangle\ \texttt{False} \leadsto^* \mathbf{return}\ \ \texttt{True}
$$

(shift-zero and dollar without answer-type modification)

Translation: EFF$\to$DEL

$$\underline{\mathsf{op}\ V} := \mathbf{S_0} k.\lambda h.h!\left(\mathbf{inj}_{\mathsf{op}}\ (\underline{V}, \{\lambda y.k!\ y\ h\})\right)$$

$$\underline{\mathbf{handle}\ M\ \mathbf{with}\ H} := \langle \underline{M} | H^{\mathrm{ret}} \rangle\ \{H^{\mathrm{ops}}\}$$

$$\begin{pmatrix} \mathbf{handle}\ M\ \mathbf{with} \\ \quad \{\mathbf{return}\ x \mapsto N_{\mathsf{ret}}\} \\ \quad \uplus \{\mathsf{op}_i\ p\ k \mapsto N_i\}_i \end{pmatrix}^{\mathrm{ret}} := x.\lambda h.\underline{N_{\mathsf{ret}}}$$

$$\begin{pmatrix} \mathbf{handle}\ M\ \mathbf{with} \\ \quad \{\mathbf{return}\ x \mapsto N_{\mathsf{ret}}\} \\ \quad \uplus \{\mathsf{op}_i\ p\ k \mapsto N_i\}_i \end{pmatrix}^{\mathrm{ops}} := \begin{array}{l} \lambda y.\mathbf{case}\ y\ \mathbf{of}\ \{ \\ \qquad (\mathbf{inj}_{\mathsf{op}_i}\ (p, k) \to \underline{N_i})_i \\ \quad \} \end{array}$$

Theorem (Correctness)

DEL *simulates* EFF *up to congruence:*

$$M \rightsquigarrow N \implies \underline{M} \rightsquigarrow^{+}_{\mathrm{cong}} \underline{N}$$

Theorem

The following macro translations do **not** *exist:*

- TYPED EFF$\to$TYPED MON *satisfying:* $M \rightsquigarrow N \implies \underline{M} \simeq \underline{N}$.
- TYPED EFF$\to$TYPED DEL *satisfying:* $M \rightsquigarrow N \implies \underline{M} \simeq \underline{N}$.

Proof sketch:

Lemma (finite denotation property)

Every closed type X denotes a **finite** *set $[\![X]\!]$.*

Take

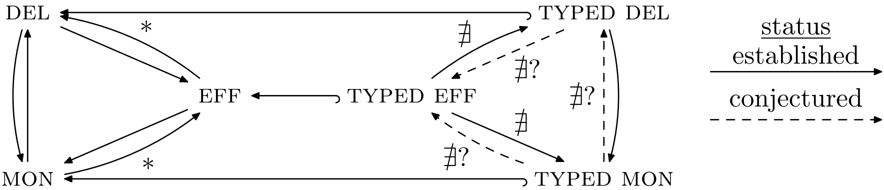
$$\mathrm{tick}^0 := \mathbf{return}\ () \qquad \mathrm{tick}^{n+1} := \mathrm{tick}(); \mathrm{tick}^n$$

and note:

$$m \neq n \implies \mathrm{tick}^n \not\simeq \mathrm{tick}^m$$

A macro translation TYPED EFF$\to$TYPED MON yields a contradiction using these two facts and MON's adequacy.

Contribution



- Syntax, operational semantics
- Macro translations
- Denotational semantics for EFF, MON
- Meta-theory:
  - Type safety
  - Adequacy and soundness for EFF, MON

- Formalisation in Abella
- Typeability preservation proofs
- Inexpressibility proofs

Expressibility must be stated as formal translations between calculi.