Handling Fibred Computational Effects

Effect Handlers in a Dependently Typed Setting

Danel Ahman

Prosecco Team at Inria Paris

HOPE 2017

September 3, 2017

Outline

- Setting the scene
    - Algebraic effects and their handlers
    - A core dependently typed effectful calculus (FoSSaCS'16)

- Why handlers + dependent types?
    - Programming with handlers + expressiveness of dep. types
    - Useful for defining predicates/types depending on computations

- Extending the FoSSaCS'16 calculus with handlers
    - Take 1: The common term-level def. of handlers (unsound)
    - Take 2: A type-level treatment of handlers

Outline

- Setting the scene
  - Algebraic effects and their handlers
  - A core dependently typed effectful calculus (FoSSaCS'16)

- Why handlers + dependent types?
  - Programming with handlers + expressiveness of dep. types
  - Useful for defining predicates/types depending on computations

- Extending the FoSSaCS'16 calculus with handlers
  - Take 1: The common term-level def. of handlers (unsound)
  - Take 2: A type-level treatment of handlers

Algebraic effects and their handlers

- Moggi taught us to model comp. effects using **monads** $(T, \eta, (-)^\dagger)$

$$\eta_A : A \to TA \qquad (f : A \to TB)^\dagger_{A,B} : TA \to TB$$

- Plotkin and Power showed that most of these monads arise from
  - **operations** – representing sources of effects

  $\text{raise} : \text{Exc} \longrightarrow 0 \qquad \text{read} : \text{Loc} \longrightarrow \text{Val} \qquad \text{write} : \text{Loc} \times \text{Val} \longrightarrow 1$

  - **equations** – describing the computational behaviour

  $\ell : \text{Loc} \mid w : 1 \vdash \text{read}_\ell\big(x.\text{write}_{\langle \ell, x \rangle}(w(\star))\big) = w(\star)$

- The algebraic approach significantly simplifies
  - **choosing** a monad/adjunction to model a given language
  - modelling **combinations** of two or more comp. effects
  - **generic** programming with effects (via handlers)

Algebraic effects and their handlers

- Moggi taught us to model comp. effects using **monads** $(T, \eta, (-)^{\dagger})$

$$\eta_A : A \to TA \qquad (f : A \to TB)^{\dagger}_{A,B} : TA \to TB$$

- Plotkin and Power showed that most of these monads arise from
  - **operations** – representing sources of effects

  $$\text{raise} : \text{Exc} \longrightarrow 0 \qquad \text{read} : \text{Loc} \longrightarrow \text{Val} \qquad \text{write} : \text{Loc} \times \text{Val} \longrightarrow 1$$

  - **equations** – describing the computational behaviour

  $$\ell : \text{Loc} \mid w : 1 \vdash \text{read}_{\ell}\big(x.\text{write}_{\langle \ell, x \rangle}\big(w(\star)\big)\big) = w(\star)$$

- The algebraic approach significantly simplifies
  - **choosing** a monad/adjunction to model a given language
  - modelling **combinations** of two or more comp. effects
  - **generic** programming with effects (via handlers)

Algebraic effects and their handlers

- Moggi taught us to model comp. effects using **monads** $(T, \eta, (-)^\dagger)$

$$\eta_A : A \to TA \qquad (f : A \to TB)^\dagger_{A,B} : TA \to TB$$

- Plotkin and Power showed that most of these monads arise from

  - **operations** – representing sources of effects

  $$\text{raise} : \text{Exc} \longrightarrow 0 \qquad \text{read} : \text{Loc} \longrightarrow \text{Val} \qquad \text{write} : \text{Loc} \times \text{Val} \longrightarrow 1$$

  - **equations** – describing the computational behaviour

  $$\ell : \text{Loc} \mid w : 1 \vdash \text{read}_\ell\big(x.\text{write}_{\langle \ell, x \rangle}\big(w(\star)\big)\big) = w(\star)$$

- The algebraic approach significantly simplifies

  - **choosing** a monad/adjunction to model a given language
  - modelling **combinations** of two or more comp. effects
  - **generic** programming with effects (via handlers)

Algebraic effects and their handlers ctd.

- Plotkin and Pretnar's **handlers** of algebraic effects
  - generalise exception handlers
  - given by redefining the given operations (they denote **algebras**)
  - example uses – rollbacks, stream redirection, concurrency, ...

- Usually included in languages using the **handling** construct

$$M \text{ handled with } \{\mathrm{op}_x(x') \mapsto N_{\mathrm{op}}\}_{\mathrm{op} \in \mathcal{S}_{\mathrm{eff}}} \text{ to } y : A \text{ in}_{\underline{C}} N_{\mathrm{ret}}$$

denoting the **homomorphism** $FA \longrightarrow \{\mathrm{op}_x(x') \mapsto N_{\mathrm{op}}\}_{\mathrm{op} \in \mathcal{S}_{\mathrm{eff}}}$

$$(\mathrm{op}_V(y.M)) \text{ handled with } \{\ldots\}_{\mathrm{op} \in \mathcal{S}_{\mathrm{eff}}} \text{ to } y : A \text{ in}_{\underline{C}} N_{\mathrm{ret}}$$
$$=$$
$$N_{\mathrm{op}}[V/x][\lambda y : O . \mathtt{thunk} (M \text{ handled with } \ldots)/x']$$

and

$$(\mathtt{return} V) \text{ handled with } \{\ldots\}_{\mathrm{op} \in \mathcal{S}_{\mathrm{eff}}} \text{ to } y : A \text{ in}_{\underline{C}} N_{\mathrm{ret}} = N_{\mathrm{ret}}[V/y]$$

Algebraic effects and their handlers ctd.

- Plotkin and Pretnar's **handlers** of algebraic effects
  - generalise exception handlers
  - given by redefining the given operations (they denote **algebras**)
  - example uses – rollbacks, stream redirection, concurrency, ...

- Usually included in languages using the **handling** construct

  M handled with $\{\text{op}_x(x') \mapsto N_{\text{op}}\}_{\text{op} \in \mathcal{S}_{\text{eff}}}$ to $y\!:\!A$ $\text{in}_{\underline{C}}$ N_{ret}

  denoting the **homomorphism** $FA \longrightarrow \{\text{op}_x(x') \mapsto N_{\text{op}}\}_{\text{op} \in \mathcal{S}_{\text{eff}}}$

  $(\text{op}_V(y.M))$ handled with $\{\ldots\}_{\text{op} \in \mathcal{S}_{\text{eff}}}$ to $y\!:\!A$ $\text{in}_{\underline{C}}$ N_{ret}
  $=$
  $N_{\text{op}}[V/x][\lambda y\!:\!O\,.\,\text{thunk}\,(M \text{ handled with } \ldots)/x']$

  and

  $(\text{return } V)$ handled with $\{\ldots\}_{\text{op} \in \mathcal{S}_{\text{eff}}}$ to $y\!:\!A$ $\text{in}_{\underline{C}}$ $N_{\text{ret}} = N_{\text{ret}}[V/y]$

Algebraic effects and their handlers ctd.

- Plotkin and Pretnar's **handlers** of algebraic effects
  - generalise exception handlers
  - given by redefining the given operations (they denote **algebras**)
  - example uses – rollbacks, stream redirection, concurrency, ...

- Usually included in languages using the **handling** construct

$$M \text{ handled with } \{\text{op}_x(x') \mapsto N_{\text{op}}\}_{\text{op} \in \mathcal{S}_{\text{eff}}} \text{ to } y : A \text{ in}_{\underline{C}} \ N_{\text{ret}}$$

  denoting the **homomorphism** $FA \longrightarrow \{\text{op}_x(x') \mapsto N_{\text{op}}\}_{\text{op} \in \mathcal{S}_{\text{eff}}}$

$$(\text{op}_V(y.M)) \text{ handled with } \{\dots\}_{\text{op} \in \mathcal{S}_{\text{eff}}} \text{ to } y : A \text{ in}_{\underline{C}} \ N_{\text{ret}}$$
$$=$$
$$N_{\text{op}}[V/x][\lambda y : O . \text{thunk} (M \text{ handled with } \dots)/x']$$

and

$$(\text{return } V) \text{ handled with } \{\dots\}_{\text{op} \in \mathcal{S}_{\text{eff}}} \text{ to } y : A \text{ in}_{\underline{C}} \ N_{\text{ret}} = N_{\text{ret}}[V/y]$$

Algebraic effects and their handlers ctd.

- Plotkin and Pretnar's **handlers** of algebraic effects
  - generalise exception handlers
  - given by redefining the given operations (they denote **algebras**)
  - example uses – rollbacks, stream redirection, concurrency, ...

- Usually included in languages using the **handling** construct

$$M \text{ handled with } \{\mathrm{op}_x(x') \mapsto N_{\mathrm{op}}\}_{\mathrm{op} \in \mathcal{S}_{\mathrm{eff}}} \text{ to } y \colon A \text{ in}_{\underline{C}} \ N_{\mathrm{ret}}$$

  denoting the **homomorphism** $FA \longrightarrow \{\mathrm{op}_x(x') \mapsto N_{\mathrm{op}}\}_{\mathrm{op} \in \mathcal{S}_{\mathrm{eff}}}$

$$(\mathrm{op}_V(y.M)) \text{ handled with } \{\ldots\}_{\mathrm{op} \in \mathcal{S}_{\mathrm{eff}}} \text{ to } y \colon A \text{ in}_{\underline{C}} \ N_{\mathrm{ret}}$$
$$=$$
$$N_{\mathrm{op}}[V/x][\lambda \, y \colon O \, . \, \mathtt{thunk} \, (M \text{ handled with } \ldots)/x']$$

  and

$$(\mathtt{return} \ V) \text{ handled with } \{\ldots\}_{\mathrm{op} \in \mathcal{S}_{\mathrm{eff}}} \text{ to } y \colon A \text{ in}_{\underline{C}} \ N_{\mathrm{ret}} \ = \ N_{\mathrm{ret}}[V/y]$$

Outline

- Setting the scene
  - Algebraic effects and their handlers
  - A core dependently typed effectful calculus (FoSSaCS'16)

- Why handlers + dependent types?
  - Programming with handlers + expressiveness of dep. types
  - Useful for defining predicates/types depending on computations

- Extending the FoSSaCS'16 calculus with handlers
  - Take 1: The common term-level def. of handlers (unsound)
  - Take 2: A type-level treatment of handlers

A core dependently typed effectful calculus

- (Model-theoretically) natural extension of type theory
  - clear distinction between **values** and **computations** (CBPV, EEC)

- **Value types** $(\Gamma \vdash A)$ and **computation types** $(\Gamma \vdash \underline{C})$

$$A, B ::= \ldots \mid U\underline{C} \qquad \underline{C}, \underline{D} ::= FA \mid \Pi x : A . \underline{C} \mid \Sigma x : A . \underline{C}$$

- **Value terms** $(\Gamma \vdash V : A)$

$$V, W ::= x \mid \ldots \mid \texttt{thunk}\, M$$

- **Computation terms** $(\Gamma \vdash M : \underline{C})$

$$M, N ::= \texttt{return}\, V \mid M \texttt{ to } x : A \texttt{ in}_{\underline{C}} N \mid \lambda x : A . M \mid M V$$
$$\mid \langle V, M \rangle \mid M \texttt{ to } (x : A, z : \underline{C}) \texttt{ in}_{\underline{D}} K \mid \texttt{force}_{\underline{C}} V$$

- **Homomorphism terms** $(\Gamma \mid z : \underline{C} \vdash K : \underline{D})$

$$K, L ::= z \mid K \texttt{ to } x : A \texttt{ in}_{\underline{C}} M \mid \ldots \qquad \text{(stacks, eval. ctxs.)}$$

A core dependently typed effectful calculus

- (Model-theoretically) natural extension of type theory
    - clear distinction between **values** and **computations** (CBPV, EEC)
- **Value types** ($\Gamma \vdash A$) and **computation types** ($\Gamma \vdash \underline{C}$)

$$A, B ::= \ldots \mid U\underline{C} \qquad \underline{C}, \underline{D} ::= FA \mid \Pi x : A . \underline{C} \mid \Sigma x : A . \underline{C}$$

- **Value terms** ($\Gamma \vdash V : A$)

    $$V, W ::= x \mid \ldots \mid \texttt{thunk } M$$

- **Computation terms** ($\Gamma \vdash M : \underline{C}$)

    $$M, N ::= \texttt{return } V \mid M \texttt{ to } x : A \texttt{ in}_{\underline{C}} N \mid \lambda x : A . M \mid M V$$
    $$\mid \langle V, M \rangle \mid M \texttt{ to } (x : A, z : \underline{C}) \texttt{ in}_{\underline{D}} K \mid \texttt{force}_{\underline{C}} V$$

- **Homomorphism terms** ($\Gamma \mid z : \underline{C} \vdash K : \underline{D}$)

    $$K, L ::= z \mid K \texttt{ to } x : A \texttt{ in}_{\underline{C}} M \mid \ldots \qquad \text{(stacks, eval. ctxs.)}$$

A core dependently typed effectful calculus

- (Model-theoretically) natural extension of type theory
  - clear distinction between **values** and **computations** (CBPV, EEC)
- **Value types** ($\Gamma \vdash A$) and **computation types** ($\Gamma \vdash \underline{C}$)

  $$A, B ::= \ \ldots \ | \ U\underline{C} \qquad \underline{C}, \underline{D} ::= F A \ | \ \Pi\, x{:}A.\, \underline{C} \ | \ \Sigma\, x{:}A.\, \underline{C}$$

- **Value terms** ($\Gamma \vdash V : A$)

  $$V, W ::= x \ | \ \ldots \ | \ \texttt{thunk}\ M$$

- **Computation terms** ($\Gamma \vdash M : \underline{C}$)

  $$M, N ::= \texttt{return}\, V \ | \ M \ \texttt{to}\ x{:}A \ \texttt{in}_{\underline{C}}\ N \ | \ \lambda x{:}A.\, M \ | \ M\, V$$
  $$| \ \langle V, M \rangle \ | \ M \ \texttt{to}\ (x{:}A, z{:}\underline{C}) \ \texttt{in}_{\underline{D}}\ K \ | \ \texttt{force}_{\underline{C}}\, V$$

- **Homomorphism terms** ($\Gamma \ | \ z{:}\underline{C} \vdash K : \underline{D}$)

  $$K, L ::= z \ | \ K \ \texttt{to}\ x{:}A \ \texttt{in}_{\underline{C}}\ M \ | \ \ldots \qquad \text{(stacks, eval. ctxs.)}$$

A core dependently typed effectful calculus

- (Model-theoretically) natural extension of type theory
  - clear distinction between **values** and **computations** (CBPV, EEC)

- **Value types** ($\Gamma \vdash A$) and **computation types** ($\Gamma \vdash \underline{C}$)

  $$A, B ::= \ldots \mid U\underline{C} \qquad \underline{C}, \underline{D} ::= FA \mid \Pi x{:}A.\underline{C} \mid \Sigma x{:}A.\underline{C}$$

- **Value terms** ($\Gamma \vdash V : A$)

  $$V, W ::= x \mid \ldots \mid \text{thunk } M$$

- **Computation terms** ($\Gamma \vdash M : \underline{C}$)

  $$M, N ::= \text{return } V \mid M \text{ to } x{:}A \text{ in}_{\underline{C}} N \mid \lambda x{:}A.M \mid M\,V$$
  $$\mid \langle V, M \rangle \mid M \text{ to } (x{:}A, z{:}\underline{C}) \text{ in}_{\underline{D}} K \mid \text{force}_{\underline{C}} V$$

- **Homomorphism terms** ($\Gamma \mid z{:}\underline{C} \vdash K : \underline{D}$)

  $$K, L ::= z \mid K \text{ to } x{:}A \text{ in}_{\underline{C}} M \mid \ldots \qquad \text{(stacks, eval. ctxs.)}$$

A core dependently typed effectful calculus

- (Model-theoretically) natural extension of type theory
  - clear distinction between **values** and **computations** (CBPV, EEC)

- **Value types** ($\Gamma \vdash A$) and **computation types** ($\Gamma \vdash \underline{C}$)

  $$A, B ::= \ \ldots \ | \ U\underline{C} \qquad \underline{C}, \underline{D} ::= F A \ | \ \Pi x : A . \underline{C} \ | \ \Sigma x : A . \underline{C}$$

- **Value terms** ($\Gamma \vdash V : A$)

  $$V, W ::= x \ | \ \ldots \ | \ \texttt{thunk } M$$

- **Computation terms** ($\Gamma \vdash M : \underline{C}$)

  $$M, N ::= \texttt{return } V \ | \ M \texttt{ to } x : A \texttt{ in}_{\underline{C}} N \ | \ \lambda x : A . M \ | \ M V$$
  $$| \ \langle V, M \rangle \ | \ M \texttt{ to } (x : A, z : \underline{C}) \texttt{ in}_{\underline{D}} K \ | \ \texttt{force}_{\underline{C}} V$$

- **Homomorphism terms** ($\Gamma \mid z : \underline{C} \vdash K : \underline{D}$)

  $$K, L ::= z \ | \ K \texttt{ to } x : A \texttt{ in}_{\underline{C}} M \ | \ \ldots \qquad \text{(stacks, eval. ctxs.)}$$

Outline

- Setting the scene

  - Algebraic effects and their handlers

  - A core dependently typed effectful calculus (FoSSaCS'16)

- Why handlers + dependent types?

  - Programming with handlers + expressiveness of dep. types

  - Useful for defining predicates/types depending on computations

- Extending the FoSSaCS'16 calculus with handlers

  - Take 1: The common term-level def. of handlers (unsound)

  - Take 2: A type-level treatment of handlers

Defining predicates on effectful comps.

- For time being, assume that we have **handlers** in the calculus

  - In particular, assume that we can also **handle into values**

    M `handled with` $\{\text{op}_x(x') \mapsto V_{\text{op}}\}_{\text{op} \in \mathcal{S}_{\text{eff}}}$ `to` $y \colon A$ `in`$_B$ V_{ret}

- Also assume that we have a Tarski-style **value universe** $\mathcal{U}$

- Then we can define **predicates** $P : UFA \to \mathcal{U}$ (a value term) by

  - equipping $\mathcal{U}$ with an **algebra** structure

  - **handling** the given computation using that algebra

  - intuitively, P (`thunk` M) computes a **proof obligation** for M

- Examples

  - **lifting predicates** from return values to (I/O)-computations

  - Dijkstra's **weakest precondition semantics** of state

  - specifying **allowed patterns** of (I/O)-computations

Defining predicates on effectful comps.

- For time being, assume that we have **handlers** in the calculus

  - In particular, assume that we can also **handle into values**

    M handled with $\{\mathrm{op}_x(x') \mapsto V_{\mathrm{op}}\}_{\mathrm{op} \in \mathcal{S}_{\mathrm{eff}}}$ to $y : A$ in$_B$ V_{ret}

- Also assume that we have a Tarski-style **value universe** $\mathcal{U}$

- Then we can define **predicates** $P : UFA \to \mathcal{U}$ (a value term) by

  - equipping $\mathcal{U}$ with an **algebra** structure

  - **handling** the given computation using that algebra

  - intuitively, P (thunk M) computes a **proof obligation** for M

- Examples

  - **lifting predicates** from return values to (I/O)-computations

  - Dijkstra's **weakest precondition semantics** of state

  - specifying **allowed patterns** of (I/O)-computations

Defining predicates on effectful comps.

- For time being, assume that we have **handlers** in the calculus

  - In particular, assume that we can also **handle into values**

    M handled with $\{\text{op}_x(x') \mapsto V_{\text{op}}\}_{\text{op} \in \mathcal{S}_{\text{eff}}}$ to $y : A$ in$_B$ V_{ret}

- Also assume that we have a Tarski-style **value universe** $\mathcal{U}$

- Then we can define **predicates** $P : UFA \to \mathcal{U}$ (a value term) by

  - equipping $\mathcal{U}$ with an **algebra** structure

  - **handling** the given computation using that algebra

  - intuitively, $P\ (\text{thunk}\ M)$ computes a **proof obligation** for M

- Examples

  - **lifting predicates** from return values to (I/O)-computations

  - Dijkstra's **weakest precondition semantics** of state

  - specifying **allowed patterns** of (I/O)-computations

Lifting predicates to effectful comps.

- Given a predicate $P : A \to \mathcal{U}$ on **return values**,

  we define a predicate $\widehat{P} : UFA \to \mathcal{U}$ on **(I/O)-comps.** as

  $\lambda y : UFA . (\texttt{force } y) \texttt{ handled with } \{\ldots\}_{\texttt{op} \in S_{\text{IO}}} \texttt{ to } y' : A \texttt{ in}_{\mathcal{U}} P y'$

  using the handler given by

  $V_{\text{read}} \overset{\text{def}}{=} \lambda y : (\Sigma x : 1 . \text{Chr} \to \mathcal{U}) . \texttt{v-pi-code}(\text{chr-code}, y' . (\texttt{snd } y) y')$

  $V_{\text{write}} \overset{\text{def}}{=} \lambda y : (\Sigma x : \text{Chr} . 1 \to \mathcal{U}) . (\texttt{snd } y) \star$

- $\widehat{P}$ is similar to the **necessity modality** from Evaluation Logic

  $\Gamma \vdash \boxdot (\widehat{P} \ (\texttt{thunk} (\text{read}^{FA}(x . \texttt{return } W)))) = \Pi x : \text{Chr} . P W$

- To get **possibility mod.**, replace $\texttt{v-pi-code}$ with $\texttt{v-sigma-code}$

Lifting predicates to effectful comps.

- Given a predicate $P : A \to \mathcal{U}$ on **return values**,

  we define a predicate $\widehat{P} : UFA \to \mathcal{U}$ on **(I/O)-comps.** as

  $$\lambda y : UFA . (\texttt{force } y) \texttt{ handled with } \{\dots\}_{op \in S_{IO}} \texttt{ to } y' : A \texttt{ in}_{\mathcal{U}} \ P \, y'$$

  using the handler given by

  $$V_{\mathsf{read}} \overset{\mathrm{def}}{=} \lambda y : (\Sigma x : 1 . \mathsf{Chr} \to \mathcal{U}) . \texttt{v-pi-code}(\texttt{chr-code}, y' . (\mathtt{snd}\, y)\, y')$$

  $$V_{\mathsf{write}} \overset{\mathrm{def}}{=} \lambda y : (\Sigma x : \mathsf{Chr} . 1 \to \mathcal{U}) . (\mathtt{snd}\, y) \star$$

- $\widehat{P}$ is similar to the **necessity modality** from Evaluation Logic

  $$\Gamma \vdash \boxdot (\widehat{P} \ (\texttt{thunk}\, (\mathtt{read}^{FA}(x . \texttt{return}\, W)))) = \Pi x : \mathsf{Chr} . \, P \, W$$

- To get **possibility mod.**, replace `v-pi-code` with `v-sigma-code`

Lifting predicates to effectful comps.

- Given a predicate $P : A \to \mathcal{U}$ on **return values**,

  we define a predicate $\widehat{P} : UFA \to \mathcal{U}$ on **(I/O)-comps.** as

  $\lambda y : UFA . (\texttt{force } y) \texttt{ handled with } \{\ldots\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{IO}}} \texttt{ to } y' : A \texttt{ in}_{\mathcal{U}} \ P \, y'$

  using the handler given by

  $$V_{\mathsf{read}} \quad \stackrel{\mathsf{def}}{=} \quad \lambda y : (\Sigma x : 1 . \mathsf{Chr} \to \mathcal{U}) . \texttt{v-pi-code}(\texttt{chr-code}, y' . (\texttt{snd } y) \, y')$$

  $$V_{\mathsf{write}} \quad \stackrel{\mathsf{def}}{=} \quad \lambda y : (\Sigma x : \mathsf{Chr} . 1 \to \mathcal{U}) . (\texttt{snd } y) \star$$

- $\widehat{P}$ is similar to the **necessity modality** from Evaluation Logic

  $\Gamma \vdash \Box(\widehat{P} \, (\texttt{thunk}(\texttt{read}^{FA}(x . \texttt{return } W)))) = \Pi x : \mathsf{Chr} . P \, W$

- To get **possibility mod.**, replace `v-pi-code` with `v-sigma-code`

Lifting predicates to effectful comps.

- Given a predicate $P : A \to \mathcal{U}$ on **return values**,

  we define a predicate $\widehat{P} : UFA \to \mathcal{U}$ on **(I/O)-comps.** as

  $$\lambda\, y : UFA \,.\, (\texttt{force}\, y)\ \texttt{handled with}\ \{\ldots\}_{\mathrm{op} \in \mathcal{S}_{\mathrm{IO}}}\ \texttt{to}\ y' : A\ \mathtt{in}_{\mathcal{U}}\ P\, y'$$

  using the handler given by

  $$V_{\mathrm{read}} \quad \stackrel{\mathrm{def}}{=} \quad \lambda\, y : (\Sigma\, x : 1 \,.\, \mathrm{Chr} \to \mathcal{U}) \,.\, \texttt{v-pi-code}\big(\texttt{chr-code}\,,\, y'.\, (\texttt{snd}\, y)\, y'\big)$$

  $$V_{\mathrm{write}} \quad \stackrel{\mathrm{def}}{=} \quad \lambda\, y : (\Sigma\, x : \mathrm{Chr} \,.\, 1 \to \mathcal{U}) \,.\, (\texttt{snd}\, y)\, \star$$

- $\widehat{P}$ is similar to the **necessity modality** from Evaluation Logic

  $$\Gamma \vdash \mathsf{El}\big(\widehat{P}\, \big(\texttt{thunk}\, (\texttt{read}^{FA}(x \,.\, \texttt{return}\, W))\big)\big) = \Pi\, x : \mathrm{Chr} \,.\, P\, W$$

- To get **possibility mod.**, replace `v-pi-code` with `v-sigma-code`

Dijkstra's weakest precondition semantics

- Given a postcondition on **return values** and **final states**

$$Q : A \to \mathsf{St} \to \mathcal{U}$$

  we define a precondition for **stateful comps.** on **initial states**

$$\mathsf{wp}_Q : UFA \to \mathsf{St} \to \mathcal{U}$$

  by

  i) handling the given comp. into a state-passing function using

$$V_{\mathsf{get}}, V_{\mathsf{put}} \quad \text{on} \quad \mathsf{St} \to (\mathcal{U} \times \mathsf{St}) \qquad \text{and} \qquad V_{\mathsf{ret}} \ "=" \ V_Q$$

  ii) feeding in the initial state, and iii) projecting out the proposition

- Then wp_Q satisfies the expected properties, e.g.,

$$\Gamma \vdash \mathsf{wp}_Q \ (\mathtt{thunk}\,(\mathtt{return}\,V)) \ = \ \lambda\, x_S{:}\mathsf{St}.\, Q\, V\, x_S \qquad : \mathsf{St} \to \mathcal{U}$$

$$\Gamma \vdash \mathsf{wp}_Q \ (\mathtt{thunk}\,(\mathtt{put}^{FA}_{V_S}(M))) \ = \ \lambda\, x_S{:}\mathsf{St}.\, \mathsf{wp}_Q \ (\mathtt{thunk}\, M)\, V_S \qquad : \mathsf{St} \to \mathcal{U}$$

Dijkstra's weakest precondition semantics

- Given a postcondition on **return values** and **final states**

$$Q : A \to \mathrm{St} \to \mathcal{U}$$

  we define a precondition for **stateful comps.** on **initial states**

$$\mathrm{wp}_Q : UFA \to \mathrm{St} \to \mathcal{U}$$

  by

  i) handling the given comp. into a state-passing function using

  $V_{\mathrm{get}}, V_{\mathrm{put}}$ on $\mathrm{St} \to (\mathcal{U} \times \mathrm{St})$ and V_{ret} "=" V_Q

  ii) feeding in the initial state, and iii) projecting out the proposition

- Then wp_Q satisfies the expected properties, e.g.,

$$\Gamma \vdash \mathrm{wp}_Q \; (\mathrm{thunk}\,(\mathtt{return}\,V)) \;=\; \lambda x_S{:}\mathrm{St}.\,Q\,V\,x_S \quad : \mathrm{St} \to \mathcal{U}$$

$$\Gamma \vdash \mathrm{wp}_Q \; (\mathrm{thunk}\,(\mathtt{put}^{FA}_{V_S}(M))) \;=\; \lambda x_S{:}\mathrm{St}.\,\mathrm{wp}_Q\,(\mathrm{thunk}\,M)\,V_S \quad : \mathrm{St} \to \mathcal{U}$$

Dijkstra's weakest precondition semantics

- Given a postcondition on **return values** and **final states**

$$Q : A \to \mathrm{St} \to \mathcal{U}$$

  we define a precondition for **stateful comps.** on **initial states**

$$\mathrm{wp}_Q : UFA \to \mathrm{St} \to \mathcal{U}$$

  by

  i) handling the given comp. into a state-passing function using

$$V_{\mathrm{get}}, V_{\mathrm{put}} \quad \text{on} \quad \mathrm{St} \to (\mathcal{U} \times \mathrm{St}) \qquad \text{and} \qquad V_{\mathrm{ret}} \; "=" \; V_Q$$

  ii) feeding in the initial state, and **iii)** projecting out the proposition

- Then wp_Q satisfies the expected properties, e.g.,

$$\Gamma \vdash \mathrm{wp}_Q \,(\mathtt{thunk}\,(\mathtt{return}\,V)) \;=\; \lambda x_S : \mathrm{St}.\, Q\, V\, x_S \qquad : \mathrm{St} \to \mathcal{U}$$

$$\Gamma \vdash \mathrm{wp}_Q \,(\mathtt{thunk}\,(\mathtt{put}^{FA}_{V_S}(M))) \;=\; \lambda x_S : \mathrm{St}.\, \mathrm{wp}_Q \,(\mathtt{thunk}\,M)\, V_S \qquad : \mathrm{St} \to \mathcal{U}$$

Dijkstra's weakest precondition semantics

- Given a postcondition on **return values** and **final states**

$$Q : A \to \mathsf{St} \to \mathcal{U}$$

we define a precondition for **stateful comps.** on **initial states**

$$\mathsf{wp}_Q : UFA \to \mathsf{St} \to \mathcal{U}$$

by

  i) handling the given comp. into a state-passing function using

$$V_{\mathrm{get}}, V_{\mathrm{put}} \quad \text{on} \quad \mathsf{St} \to (\mathcal{U} \times \mathsf{St}) \qquad \text{and} \qquad V_{\mathrm{ret}} \text{ ''='' } V_Q$$

  ii) feeding in the initial state, and **iii)** projecting out the proposition

- Then wp_Q satisfies the expected properties, e.g.,

$$\Gamma \vdash \mathsf{wp}_Q\,(\mathrm{thunk}\,(\mathtt{return}\,V)) \;\; = \;\; \lambda\,x_S\!:\!\mathsf{St}\,.\,Q\;V\;x_S \qquad\qquad : \mathsf{St} \to \mathcal{U}$$

$$\Gamma \vdash \mathsf{wp}_Q\,(\mathrm{thunk}\,(\mathtt{put}^{FA}_{V_S}(M))) \;\; = \;\; \lambda\,x_S\!:\!\mathsf{St}\,.\,\mathsf{wp}_Q\,(\mathrm{thunk}\,M)\;V_S \quad : \mathsf{St} \to \mathcal{U}$$

Specifying allowed patterns of I/O-effects

- We assume an **inductive type** Protocol, given by

$$e : \text{Protocol} \qquad r : (\text{Chr} \to \text{Protocol}) \to \text{Protocol}$$

$$w : (\text{Chr} \to \mathcal{U}) \times \text{Protocol} \to \text{Protocol}$$

- Given a **protocol** Pr : Protocol, we define

$$\widehat{\text{Pr}} : UFA \to \mathcal{U}$$

by handling a given comp. using

$$V_{\text{read}}, V_{\text{write}} \qquad \text{on} \qquad \text{Protocol} \to \mathcal{U}$$

where

$$V_{\text{read}} \ \langle V, V_{\text{rk}} \rangle \ (r \ \text{Pr}') \quad \overset{\text{def}}{=} \quad \texttt{v-pi-code}(\texttt{chr-code}, y \,.\, (V_{\text{rk}} \ y)(\text{Pr}' \ y))$$

$$V_{\text{write}} \ \langle V, V_{\text{wk}} \rangle \ (w \ \langle P, \text{Pr}' \rangle) \quad \overset{\text{def}}{=} \quad \texttt{v-sigma-code}(P \ V, y \,.\, V_{\text{wk}} \star \text{Pr}')$$

$$\_ \quad \overset{\text{def}}{=} \quad \texttt{empty-code}$$

Specifying allowed patterns of I/O-effects

- We assume an **inductive type** Protocol, given by

$$\texttt{e} : \text{Protocol} \qquad \texttt{r} : (\text{Chr} \to \text{Protocol}) \to \text{Protocol}$$

$$\texttt{w} : (\text{Chr} \to \mathcal{U}) \times \text{Protocol} \to \text{Protocol}$$

- Given a **protocol** Pr : Protocol, we define

$$\widehat{\text{Pr}} : UFA \to \mathcal{U}$$

by handling a given comp. using

$$V_{\text{read}}, V_{\text{write}} \qquad \text{on} \qquad \text{Protocol} \to \mathcal{U}$$

where

$$V_{\text{read}} \, \langle V, V_{\text{rk}} \rangle \, (\texttt{r} \, \text{Pr}') \quad \overset{\text{def}}{=} \quad \texttt{v-pi-code}(\texttt{chr-code}, y . (V_{\text{rk}} \, y)(\text{Pr}' \, y))$$

$$V_{\text{write}} \, \langle V, V_{\text{wk}} \rangle \, (\texttt{w} \, \langle P . \text{Pr}' \rangle) \quad \overset{\text{def}}{=} \quad \texttt{v-sigma-code}(P \, V, y . V_{\text{wk}} \star \text{Pr}')$$

$$\_ \qquad \overset{\text{def}}{=} \quad \texttt{empty-code}$$

Specifying allowed patterns of I/O-effects

- We assume an **inductive type** Protocol, given by

$$\texttt{e} : \text{Protocol} \qquad \texttt{r} : (\text{Chr} \rightarrow \text{Protocol}) \rightarrow \text{Protocol}$$

$$\texttt{w} : (\text{Chr} \rightarrow \mathcal{U}) \times \text{Protocol} \rightarrow \text{Protocol}$$

- Given a **protocol** Pr : Protocol, we define

$$\widehat{\text{Pr}} : \mathit{UFA} \rightarrow \mathcal{U}$$

by handling a given comp. using

$$V_{\text{read}}, V_{\text{write}} \qquad \text{on} \qquad \text{Protocol} \rightarrow \mathcal{U}$$

where

$$V_{\text{read}} \ \langle V, V_{\text{rk}} \rangle \ (\texttt{r} \ \text{Pr}') \quad \overset{\text{def}}{=} \quad \texttt{v-pi-code}(\texttt{chr-code}, y \, . \, (V_{\text{rk}} \, y)(\text{Pr}' \, y))$$

$$V_{\text{write}} \ \langle V, V_{\text{wk}} \rangle \ (\texttt{w} \ \langle P, \text{Pr}' \rangle) \quad \overset{\text{def}}{=} \quad \texttt{v-sigma-code}(P \, V, y \, . \, V_{\text{wk}} \star \text{Pr}')$$

$$\_ \quad \overset{\text{def}}{=} \quad \texttt{empty-code}$$

Specifying allowed patterns of I/O-effects

- We assume an **inductive type** Protocol, given by

$$\texttt{e} : \text{Protocol} \qquad \texttt{r} : (\text{Chr} \to \text{Protocol}) \to \text{Protocol}$$

$$\texttt{w} : (\text{Chr} \to \mathcal{U}) \times \text{Protocol} \to \text{Protocol}$$

- Given a **protocol** Pr : Protocol, we define

$$\widehat{\text{Pr}} : \mathit{UFA} \to \mathcal{U}$$

by handling a given comp. using

$$V_{\text{read}}, V_{\text{write}} \qquad \text{on} \qquad \text{Protocol} \to \mathcal{U}$$

where

$$V_{\text{read}} \ \langle V, V_{\text{rk}} \rangle \ (\texttt{r}\ \text{Pr}') \ \overset{\text{def}}{=} \ \texttt{v-pi-code}(\texttt{chr-code}, y\,.\,(V_{\text{rk}}\ y)(\text{Pr}'\ y))$$

$$V_{\text{write}} \ \langle V, V_{\text{wk}} \rangle \ (\texttt{w}\ \langle P, \text{Pr}' \rangle) \ \overset{\text{def}}{=} \ \texttt{v-sigma-code}(P\ V, y\,.\,V_{\text{wk}} \star \text{Pr}')$$

$$\_ \ \overset{\text{def}}{=} \ \texttt{empty-code}$$

Specifying allowed patterns of I/O-effects

- We assume an **inductive type** Protocol, given by

$$\texttt{e} : \text{Protocol} \qquad \texttt{r} : (\text{Chr} \to \text{Protocol}) \to \text{Protocol}$$

$$\texttt{w} : (\text{Chr} \to \mathcal{U}) \times \text{Protocol} \to \text{Protocol}$$

- Given a **protocol** Pr : Protocol, we define

$$\widehat{\text{Pr}} : \mathit{UFA} \to \mathcal{U}$$

  by handling a given comp. using

$$V_{\text{read}}, V_{\text{write}} \qquad \text{on} \qquad \text{Protocol} \to \mathcal{U}$$

  where

$$V_{\text{read}} \ \langle V, V_{\text{rk}} \rangle \ (\texttt{r} \ \text{Pr}') \quad \overset{\text{def}}{=} \quad \texttt{v-pi-code}\big(\texttt{chr-code}, y \,.\, (V_{\text{rk}} \, y) \, (\text{Pr}' \, y)\big)$$

$$V_{\text{write}} \ \langle V, V_{\text{wk}} \rangle \ (\texttt{w} \ \langle P, \text{Pr}' \rangle) \quad \overset{\text{def}}{=} \quad \texttt{v-sigma-code}\big(P \, V, y \,.\, V_{\text{wk}} \star \text{Pr}'\big)$$

$$\_ \qquad \qquad \overset{\text{def}}{=} \quad \texttt{empty-code}$$

Outline

- Setting the scene
  - Algebraic effects and their handlers
  - A core dependently typed effectful calculus (FoSSaCS'16)

- Why handlers + dependent types?
  - Programming with handlers + expressiveness of dep. types
  - Useful for defining predicates/types depending on computations

- Extending the FoSSaCS'16 calculus with handlers
  - Take 1: The common term-level def. of handlers (unsound)
  - Take 2: A type-level treatment of handlers

Fibred algebraic effects

- To include fib. alg. effects $(\mathcal{S}_{\text{eff}}, \mathcal{E}_{\text{eff}})$ in our calculus, we

  - extend its computation terms with **algebraic operations**

$$\frac{\Gamma \vdash V : I \quad \Gamma \vdash \underline{C} \quad \Gamma, y : O[V/x] \vdash M : \underline{C}}{\Gamma \vdash \text{op}_V^{\underline{C}}(y.M) : \underline{C}}$$

  for every operation symbol $\text{op} : (x : I) \longrightarrow O$ in $\mathcal{S}_{\text{eff}}$

  - include **equations** $\Gamma \mid \Delta \vdash T_1 = T_2$ in $\mathcal{E}_{\text{eff}}$

  - include a general **algebraicity equation**

$$\frac{\Gamma \mid z : \underline{C} \vdash K : \underline{D} \quad \Gamma \vdash V : I \quad \Gamma, y : O[V/x] \vdash M : \underline{C}}{\Gamma \vdash K[\text{op}_V^{\underline{C}}(y.M)/z] = \text{op}_V^{\underline{D}}(y.K[M/z]) : \underline{D}}$$

Fibred algebraic effects

- To include fib. alg. effects $(\mathcal{S}_{\text{eff}}, \mathcal{E}_{\text{eff}})$ in our calculus, we

  - extend its computation terms with **algebraic operations**

  $$\frac{\Gamma \vdash V : I \quad \Gamma \vdash \underline{C} \quad \Gamma, y : O[V/x] \vdash M : \underline{C}}{\Gamma \vdash \mathrm{op}_V^{\underline{C}}(y.M) : \underline{C}}$$

  for every operation symbol $\mathrm{op} : (x : I) \longrightarrow O$ in $\mathcal{S}_{\text{eff}}$

  - include **equations** $\Gamma \mid \Delta \vdash T_1 = T_2$ in $\mathcal{E}_{\text{eff}}$

  - include a general **algebraicity equation**

  $$\frac{\Gamma \mid z : \underline{C} \vdash K : \underline{D} \quad \Gamma \vdash V : I \quad \Gamma, y : O[V/x] \vdash M : \underline{C}}{\Gamma \vdash K[\mathrm{op}_V^{\underline{C}}(y.M)/z] = \mathrm{op}_V^{\underline{D}}(y.K[M/z]) : \underline{D}}$$

Fibred algebraic effects

- To include fib. alg. effects $(\mathcal{S}_{\text{eff}}, \mathcal{E}_{\text{eff}})$ in our calculus, we

  - extend its computation terms with **algebraic operations**

  $$\frac{\Gamma \vdash V : I \quad \Gamma \vdash \underline{C} \quad \Gamma, y : O[V/x] \vdash M : \underline{C}}{\Gamma \vdash \text{op}_V^{\underline{C}}(y.M) : \underline{C}}$$

  for every operation symbol $\text{op} : (x : I) \longrightarrow O$ in $\mathcal{S}_{\text{eff}}$

  - include **equations** $\Gamma \mid \Delta \vdash T_1 = T_2$ in $\mathcal{E}_{\text{eff}}$

  - include a general **algebraicity equation**

  $$\frac{\Gamma \mid z : \underline{C} \vdash K : \underline{D} \quad \Gamma \vdash V : I \quad \Gamma, y : O[V/x] \vdash M : \underline{C}}{\Gamma \vdash K[\text{op}_V^{\underline{C}}(y.M)/z] = \text{op}_V^{\underline{D}}(y.K[M/z]) : \underline{D}}$$

Handlers for fibred algebraic effects

- **Take 1:** Let's use their conventional term-level definition

  - include the handling construct for **computation terms**

    M handled with $\{op_x(x') \mapsto N_{op}\}_{op \in S_{eff}}$ to $y : A$ in$_{\underline{C}}$ N_{ret}

  - as handling denotes a homomorphism, also for **hom. terms**

    K handled with $\{op_x(x') \mapsto N_{op}\}_{op \in S_{eff}}$ to $y : A$ in$_{\underline{C}}$ N_{ret}

  - but then we can prove the **unsound equation**

    $$\Gamma \vdash write_a^{F1}(return \star) = write_z^{F1}(return \star) : F1$$

    by **handling**

    $$write_a^{F1}(return \star)$$

    with

    $$write_x(x') \mapsto write_z(force\,(x'\,\star))$$

    and exploiting a non-convergent **critical pair** in the eq. theory

Handlers for fibred algebraic effects

- **Take 1:** Let's use their conventional term-level definition

  - include the handling construct for **computation terms**

    M handled with $\{\mathrm{op}_x(x') \mapsto N_{\mathrm{op}}\}_{\mathrm{op} \in \mathcal{S}_{\mathrm{eff}}}$ to $y : A$ in$_{\underline{C}}$ N_{ret}

  - as handling denotes a homomorphism, also for **hom. terms**

    K handled with $\{\mathrm{op}_x(x') \mapsto N_{\mathrm{op}}\}_{\mathrm{op} \in \mathcal{S}_{\mathrm{eff}}}$ to $y : A$ in$_{\underline{C}}$ N_{ret}

  - but then we can prove the **unsound equation**

    $$\Gamma \vdash \mathrm{write}_a^{F1}(\mathrm{return}\,\star) = \mathrm{write}_z^{F1}(\mathrm{return}\,\star) : F1$$

    by **handling**

    $$\mathrm{write}_a^{F1}(\mathrm{return}\,\star)$$

    with

    $$\mathrm{write}_x(x') \mapsto \mathrm{write}_z(\mathrm{force}\,(x'\,\star))$$

    and exploiting a non-convergent **critical pair** in the eq. theory

Handlers for fibred algebraic effects

- **Take 1:** Let's use their conventional term-level definition

  - include the handling construct for **computation terms**

    M `handled with` $\{\mathtt{op}_x(x') \mapsto N_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}}$ `to` $y\!:\!A$ `in`$_{\underline{C}}$ N_{ret}

  - as handling denotes a homomorphism, also for **hom. terms**

    K `handled with` $\{\mathtt{op}_x(x') \mapsto N_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}}$ `to` $y\!:\!A$ `in`$_{\underline{C}}$ N_{ret}

  - but then we can prove the **unsound equation**

    $\Gamma \vdash \mathtt{write}_a^{F1}(\mathtt{return}\star) = \mathtt{write}_z^{F1}(\mathtt{return}\star) : F1$

    by **handling**

    $\mathtt{write}_a^{F1}(\mathtt{return}\star)$

    with

    $\mathtt{write}_x(x') \mapsto \mathtt{write}_z(\mathtt{force}\,(x'\star))$

    and exploiting a non-convergent **critical pair** in the eq. theory

Handlers for fibred algebraic effects

- **Take 1:** Let's use their conventional term-level definition

    - include the handling construct for **computation terms**

        M `handled with` $\{\mathtt{op}_x(x') \mapsto N_{\mathtt{op}}\}_{\mathtt{op} \in \mathcal{S}_{\mathsf{eff}}}$ `to` $y : A$ `in`$_{\underline{C}}$ N_{ret}

    - as handling denotes a homomorphism, also for **hom. terms**

        K `handled with` $\{\mathtt{op}_x(x') \mapsto N_{\mathtt{op}}\}_{\mathtt{op} \in \mathcal{S}_{\mathsf{eff}}}$ `to` $y : A$ `in`$_{\underline{C}}$ N_{ret}

    - but then we can prove the **unsound equation**

        $$\Gamma \vdash \mathtt{write}_{\mathbf{a}}^{F1}(\mathtt{return}\,\star) = \mathtt{write}_{\mathbf{z}}^{F1}(\mathtt{return}\,\star) : F1$$

        by **handling**

        $$\mathtt{write}_{\mathbf{a}}^{F1}(\mathtt{return}\,\star)$$

        with

        $$\mathtt{write}_x(x') \mapsto \mathtt{write}_z(\mathtt{force}\,(x'\,\star))$$

        and exploiting a non-convergent **critical pair** in the eq. theory

Handlers for fibred algebraic effects

- **Take 1:** Let's use their conventional term-level definition

  - include the handling construct for **computation terms**

    M `handled with` $\{op_x(x') \mapsto N_{op}\}_{op \in S_{eff}}$ `to` $y : A$ `in`$_{\underline{C}}$ N_{ret}

  - as handling denotes a homomorphism, also for **hom. terms**

    K `handled with` $\{op_x(x') \mapsto N_{op}\}_{op \in S_{eff}}$ `to` $y : A$ `in`$_{\underline{C}}$ N_{ret}

  - but then we can prove the **unsound equation**

    $$\Gamma \vdash \mathtt{write}_a^{F1}(\mathtt{return}\,\star) = \mathtt{write}_z^{F1}(\mathtt{return}\,\star) : F1$$

    by **handling**

    $$\mathtt{write}_a^{F1}(\mathtt{return}\,\star)$$

    with

    $$\mathtt{write}_x(x') \mapsto \mathtt{write}_z(\mathtt{force}\,(x'\,\star))$$

    and exploiting a non-convergent **critical pair** in the eq. theory

Handlers for fibred algebraic effects ctd.

- Possible ways to solve this unsoundness problem

  - **Option 1:** Change the FoSSaCS'16 calculus

    - change the equational theory of homomorphism terms

    - hom. terms wouldn't denote homomorphisms any more

    - investigated for exceptions in CBPV with stacks in [Levy'06]

  - **Option 2:** Keep the FoSSaCS'16 calculus unchanged

    - extend it so that handling for comp. terms is derivable

    - while making sure that the calculus remains sound

    - **key idea:** comp. types and handlers both denote algebras

Handlers for fibred algebraic effects ctd.

- Possible ways to solve this unsoundness problem

    - **Option 1:** Change the FoSSaCS'16 calculus
        - change the equational theory of homomorphism terms
        - hom. terms wouldn't denote homomorphisms any more
        - investigated for exceptions in CBPV with stacks in [Levy'06]

    - **Option 2:** Keep the FoSSaCS'16 calculus unchanged
        - extend it so that handling for comp. terms is derivable
        - while making sure that the calculus remains sound
        - **key idea:** comp. types and handlers both denote algebras

Handlers for fibred algebraic effects ctd.

- Possible ways to solve this unsoundness problem

    - **Option 1:** Change the FoSSaCS'16 calculus
        - change the equational theory of homomorphism terms
        - hom. terms wouldn't denote homomorphisms any more
        - investigated for exceptions in CBPV with stacks in [Levy'06]

    - **Option 2:** Keep the FoSSaCS'16 calculus unchanged
        - extend it so that handling for comp. terms is derivable
        - while making sure that the calculus remains sound
        - **key idea:** comp. types and handlers both denote algebras

Handlers for fibred algebraic effects ctd.

- **Take 2:** A type-based treatment of handlers

  - we introduce the **user-defined algebra type** (comp. type)

    $$\frac{\Gamma \vdash A \quad \{\Gamma \vdash V_{op} : (\Sigma x{:}I.O \to A) \to A\}_{op \in \mathcal{S}_{eff}}}{\Gamma \vdash \langle A, \{V_{op}\}_{op \in \mathcal{S}_{eff}} \rangle}$$

  - we introduce corresponding **elimination forms**

    $$\frac{\Gamma \vdash M : \langle A, \{V_{op}\}_{op \in \mathcal{S}_{eff}} \rangle \quad \Gamma \vdash \underline{C} \quad \Gamma, x{:}A \vdash N : \underline{C}}{\Gamma \vdash M \text{ as } x{:}A \text{ in } N : \underline{C}}$$

    and

    $$\frac{\cdots}{\Gamma \mid z{:}\underline{C} \vdash K \text{ as } x{:}A \text{ in } N : \underline{D}}$$

Handlers for fibred algebraic effects ctd.

- **Take 2:** A type-based treatment of handlers

  - we introduce the **user-defined algebra type** (comp. type)

  $$\frac{\Gamma \vdash A \quad \{\Gamma \vdash V_{\mathsf{op}} : (\Sigma x : I.O \to A) \to A\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \quad V_{\mathsf{op}} \text{ satisfy the equations in } \mathcal{E}_{\mathsf{eff}}}{\Gamma \vdash \langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle}$$

  - we introduce corresponding **elimination forms**

  $$\frac{\Gamma \vdash M : \langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle \quad \Gamma \vdash \underline{C} \quad \Gamma, x : A \vdash N : \underline{C} \quad N \text{ behaves as a homomorphism in } x \text{ (i.e., commutes with ops.)}}{\Gamma \vdash M \text{ as } x : A \text{ in } N : \underline{C}}$$

  and

  $$\frac{\dots}{\Gamma \mid z : \underline{C} \vdash K \text{ as } x : A \text{ in } N : \underline{D}}$$

Handlers for fibred algebraic effects ctd.

- **Take 2:** A type-based treatment of handlers

  - we introduce the **user-defined algebra type** (comp. type)

  $$\frac{\Gamma \vdash A \quad \{\Gamma \vdash V_{\mathsf{op}} : (\Sigma x : I . O \to A) \to A\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \quad V_{\mathsf{op}} \text{ satisfy the equations in } \mathcal{E}_{\mathsf{eff}}}{\Gamma \vdash \langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle}$$

  - we introduce corresponding **elimination forms**

  $$\frac{\Gamma \vdash M : \langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle \quad \Gamma \vdash \underline{C} \quad \Gamma, x : A \vdash N : \underline{C} \quad N \text{ behaves as a homomorphism in x (i.e., commutes with ops.)}}{\Gamma \vdash M \text{ as } x : A \text{ in } N : \underline{C}}$$

  and

  $$\frac{\dots}{\Gamma \mid z : \underline{C} \vdash K \text{ as } x : A \text{ in } N : \underline{D}}$$

Handlers for fibred algebraic effects ctd.

- **Take 2:** A type-based treatment of handlers

  - we introduce the **user-defined algebra type** (comp. type)

    $$\frac{\Gamma \vdash A \quad \{\Gamma \vdash V_{\mathsf{op}} : (\Sigma x : I.O \to A) \to A\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}}}{\Gamma \vdash \langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle} \\ V_{\mathsf{op}} \text{ satisfy the equations in } \mathcal{E}_{\mathsf{eff}}$$

  - we introduce corresponding **elimination forms**

    $$\frac{\Gamma \vdash M : \langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle \quad \Gamma \vdash \underline{C} \quad \Gamma, x : A \vdash N : \underline{C}}{\Gamma \vdash M \text{ as } x : A \text{ in } N : \underline{C}} \\ N \text{ behaves as a homomorphism in } x \text{ (i.e., commutes with ops.)}$$

    and

    $$\frac{\ldots}{\Gamma \mid z : \underline{C} \vdash K \text{ as } x : A \text{ in } N : \underline{D}}$$

Handlers for fibred algebraic effects ctd.

- **Take 2:** A type-based treatment of handlers

  - extend the equational theory of **value types** with

    $$\Gamma \vdash U \langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle = A$$

    (what about the corresponding η-equation for comp. types?)

  - extend the equational theory of **comp.** and **hom. terms** with

    $$\Gamma \vdash (\mathtt{force}_{\langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle} V) \text{ as } x \colon A \text{ in } N = N[V/x] : \underline{C}$$

    $$\Gamma \vdash M \text{ as } x \colon A \text{ in } K[\mathtt{force}_{\langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle} x/z] = K[M/z] : \underline{C}$$

    $$\Gamma \vdash \mathtt{op}_V^{\langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle}(y.M)$$
    $$= \mathtt{force}_{\langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle} (V_{\mathsf{op}} \langle V, \lambda\, y.\mathtt{thunk}\, M \rangle) : \langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle$$

Handlers for fibred algebraic effects ctd.

- **Take 2:** A type-based treatment of handlers

  - extend the equational theory of **value types** with

    $$\Gamma \vdash U \langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle = A$$

    (what about the corresponding η-equation for comp. types?)

  - extend the equational theory of **comp.** and **hom. terms** with

    $$\Gamma \vdash (\mathtt{force}_{\langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle} V) \text{ as } x \colon A \text{ in } N = N[V/x] : \underline{C}$$

    $$\Gamma \vdash M \text{ as } x \colon A \text{ in } K[\mathtt{force}_{\langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle} x/z] = K[M/z] : \underline{C}$$

    $$\Gamma \vdash \mathsf{op}_V^{\langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle}(y.M)$$
    $$= \mathtt{force}_{\langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle} (V_{\mathsf{op}} \langle V, \lambda y. \mathtt{thunk}\, M \rangle) : \langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle$$

Handlers for fibred algebraic effects ctd.

- **Take 2:** A type-based treatment of handlers

  - extend the equational theory of **value types** with

    $$\Gamma \vdash U \langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle = A$$

    (what about the corresponding η-equation for comp. types?)

  - extend the equational theory of **comp.** and **hom. terms** with

    $$\Gamma \vdash (\mathtt{force}_{\langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle} V) \text{ as } x\!:\!A \text{ in } N = N[V/x] : \underline{C}$$

    $$\Gamma \vdash M \text{ as } x\!:\!A \text{ in } K[\mathtt{force}_{\langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle} \, x/z] = K[M/z] : \underline{C}$$

$$\Gamma \vdash \mathsf{op}_V^{\langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle}(y.M)$$
$$= \mathtt{force}_{\langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle} (V_{\mathsf{op}} \langle V, \lambda \, y.\mathtt{thunk}\, M \rangle) : \langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle$$

Handlers for fibred algebraic effects ctd.

- **Take 2:** A type-based treatment of handlers

  - we can then routinely derive the **handling construct**

    M `handled with` $\{\text{op}_x(x') \mapsto N_{\text{op}}\}_{\text{op} \in \mathcal{S}_{\text{eff}}}$ `to` $y\!:\!A$ `in`$_{\underline{C}}$ N_{ret}

    using **sequential composition**, thunking, and forcing:

    $\text{force}_{\underline{C}}\left(\text{thunk}\left(\underbrace{M \text{ to } y\!:\!A \text{ in } (\text{force}_{(U\underline{C} \sqsubseteq \{V_{\text{op}}\}_{\text{op} \in \mathcal{S}_{\text{eff}}})}(\text{thunk } N_{\text{ret}}))}_{\text{has type } (U\underline{C} \sqsubseteq \{V_{\text{op}}\}_{\text{op} \in \mathcal{S}_{\text{eff}}})}\right)\right)$

    where the value terms V_{op} are derived from the comp. terms N_{op}

    - satisfies the standard β-equations for handling

    - **handling into values** can be derived analogously

Handlers for fibred algebraic effects ctd.

- **Take 2:** A type-based treatment of handlers

  - we can then routinely derive the **handling construct**

    $$M \text{ handled with } \{op_x(x') \mapsto N_{op}\}_{op \, \in \, \mathcal{S}_{\mathsf{eff}}} \text{ to } y \colon A \text{ in}_{\underline{C}} \ N_{\mathsf{ret}}$$

    using **sequential composition**, thunking, and forcing:

    $$\texttt{force}_{\underline{C}} \left(\texttt{thunk} \left(\underbrace{M \text{ to } y \colon A \text{ in } \left(\texttt{force}_{\langle U\underline{C}, \{V_{op}\}_{op \in \mathcal{S}_{\mathsf{eff}}} \rangle} \left(\texttt{thunk } N_{\mathsf{ret}} \right) \right)}_{\text{has type } \langle U\underline{C}, \{V_{op}\}_{op \in \mathcal{S}_{\mathsf{eff}}} \rangle} \right) \right)$$

    where the value terms V_{op} are derived from the comp. terms N_{op}

  - satisfies the standard β-equations for handling

  - **handling into values** can be derived analogously

Handlers for fibred algebraic effects ctd.

- **Take 2:** A type-based treatment of handlers

  - we can then routinely derive the **handling construct**

    $$M \texttt{ handled with } \{\text{op}_x(x') \mapsto N_{\text{op}}\}_{\text{op} \in \mathcal{S}_{\text{eff}}} \texttt{ to } y\!:\!A \texttt{ in}_{\underline{C}} \; N_{\text{ret}}$$

    using **sequential composition**, thunking, and forcing:

    $$\texttt{force}_{\underline{C}} \big(\texttt{thunk} \big(\underbrace{M \texttt{ to } y\!:\!A \texttt{ in } \big(\texttt{force}_{\langle U\underline{C}, \{V_{\text{op}}\}_{\text{op} \in \mathcal{S}_{\text{eff}}} \rangle} \, (\texttt{thunk } N_{\text{ret}}) \big)}_{\text{has type } \langle U\underline{C}, \{V_{\text{op}}\}_{\text{op} \in \mathcal{S}_{\text{eff}}} \rangle} \big) \big)$$

    where the value terms V_{op} are derived from the comp. terms N_{op}

  - satisfies the standard β-equations for handling

  - **handling into values** can be derived analogously

Conclusion

- In this talk, we saw that
  - handlers are useful for defining preds./types on computations
    - more generally, homomorphic type dep. on comps. is natural
    - this naturality was also observed in [Pédrot, Tabareau'17]
  - unsoundness problems can arise when accommodating handlers
    - handlers defined at term-level, while denoting algebras
  - handlers admit a principled type-based treatment
    - conventional term-level def. is derivable using seq. comp.

- Future work includes
  - general account of defining predicates on alg. effects
  - operational semantics (complex values $+$ eq. for ops.)
  - presentations of the calculus without hom. terms (eq. proof obl.)

Thank you!

Questions?