Fibred Computational Effects

(thesis overview)

Danel Ahman

(Prosecco team, INRIA Paris)

June 29, 2017

Overall aims

Investigate language design principles for combining

- refinement types $(\langle \mathrm{op} \rangle(\mathcal{I}_1, \ldots, \mathcal{I}_n), \ldots)$
- dependent types $(\Pi, \Sigma, V =_A W, \ldots)$
- computational effects (state, I/O, probability, recursion, ...)

The goal of this work was to establish that

- refinement types and comp. effects admit a natural combination
- dependent types and comp. effects admit a natural combination
- this combination covers a wide range of computational effects

The work was driven by two guiding questions

- Should one allow effectful programs in types?
- How should one treat type-dependency in sequential composition?

Thesis structure

- **Chapter 1:** Introduction and related work
- **Chapter 2:** Preliminaries of models of effects and dep. types
- **Chapter 3:** The core language eMLTT
- **Chapter 4:** Categorical models of eMLTT
- **Chapter 5:** Interpretation of eMLTT, soundness, completeness
- **Chapter 6:** An extension of eMLTT with algebraic effects
- **Chapter 7:** An extension of eMLTT with handlers
- **Chapter 8:** Conclusion and future work
- **Appendices:** Details of longer proofs

Chapter 3: The core language eMLTT

Values and computations

- clear distinction, at the level of types and terms (CBPV, EEC)
- both kinds of types are allowed to depend only on values
- selective dependency on computations is possible via thunks

Value and computation types

$$A, B \quad ::= \quad \ldots \mid \Sigma x \!:\! A.B \mid \Pi x \!:\! A.B \mid V =_A W \mid U\underline{C} \mid \underline{C} \multimap \underline{D}$$

$$\underline{C}, \underline{D} \quad ::= \quad FA \mid \Sigma x \!:\! A.\underline{C} \mid \Pi x \!:\! A.\underline{C}$$

Value, computation, and homomorphism terms

$$V, W \quad ::= \quad x \mid \ldots \mid \mathtt{thunk}\, M \mid \lambda z \!:\! \underline{C}.K$$

$$M, N \quad ::= \quad \mathtt{force}_{\underline{C}}\, V \mid \mathtt{return}\, V \mid M \;\mathtt{to}\; x \!:\! A \;\mathtt{in}_{\underline{C}}\; N \mid \langle V, M \rangle \mid M \;\mathtt{to}\; (x \!:\! A, z \!:\! \underline{C}) \;\mathtt{in}_{\underline{D}}\; K \mid \ldots$$

$$K, L \quad ::= \quad z \mid K \;\mathtt{to}\; x \!:\! A \;\mathtt{in}_{\underline{C}}\; M \mid \langle V, K \rangle \mid \ldots$$

Chapter 3: The core language eMLTT

Values and computations

- clear distinction, at the level of types and terms (CBPV, EEC)
- both kinds of types are allowed to depend only on values
- selective dependency on computations is possible via thunks

Value and computation types

$$A, B \quad ::= \quad \ldots \mid \Sigma x{:}A.B \mid \Pi x{:}A.B \mid V =_A W \mid U\underline{C} \mid \underline{C} \multimap \underline{D}$$

$$\underline{C}, \underline{D} \quad ::= \quad FA \mid \Sigma x{:}A.\underline{C} \mid \Pi x{:}A.\underline{C}$$

Value, computation, and homomorphism terms

$$V, W \quad ::= \quad x \mid \ldots \mid \mathtt{thunk}\, M \mid \lambda z{:}\underline{C}.K$$

$$M, N \quad ::= \quad \mathtt{force}_{\underline{C}}\, V \mid \mathtt{return}\, V \mid M \;\mathtt{to}\; x{:}A \;\mathtt{in}_{\underline{C}}\; N$$

$$\mid \quad \langle V, M \rangle \quad \mid M \;\mathtt{to}\; (x{:}A, z{:}\underline{C}) \;\mathtt{in}_{\underline{D}}\; K \mid \ldots$$

$$K, L \quad ::= \quad z \mid K \;\mathtt{to}\; x{:}A \;\mathtt{in}_{\underline{C}}\; M \mid \langle V, K \rangle \mid \ldots$$

Chapter 3: The core language eMLTT

Values and computations

- clear distinction, at the level of types and terms (CBPV, EEC)
- both kinds of types are allowed to depend only on values
- selective dependency on computations is possible via thunks

Value and computation types

$$A, B \quad ::= \quad \ldots \mid \Sigma x\!:\!A.B \mid \Pi x\!:\!A.B \mid V =_A W \mid U\underline{C} \mid \underline{C} \multimap \underline{D}$$

$$\underline{C}, \underline{D} \quad ::= \quad FA \mid \Sigma x\!:\!A.\underline{C} \mid \Pi x\!:\!A.\underline{C}$$

Value, computation, and homomorphism terms

$$V, W \quad ::= \quad x \mid \ldots \mid \mathtt{thunk}\, M \mid \lambda z\!:\!\underline{C}.K$$

$$M, N \quad ::= \quad \mathtt{force}_{\underline{C}}\, V \mid \mathtt{return}\, V \mid M \;\mathtt{to}\; x\!:\!A \;\mathtt{in}_{\underline{C}}\; N$$

$$\mid \quad \langle V, M \rangle \quad \mid M \;\mathtt{to}\; (x\!:\!A, z\!:\!\underline{C}) \;\mathtt{in}_{\underline{D}}\; K \mid \ldots$$

$$K, L \quad ::= \quad z \mid K \;\mathtt{to}\; x\!:\!A \;\mathtt{in}_{\underline{C}}\; M \mid \langle V, K \rangle \mid \ldots$$

Chapter 3: The core language eMLTT

Values and computations

- clear distinction, at the level of types and terms (CBPV, EEC)
- both kinds of types are allowed to depend only on values
- selective dependency on computations is possible via thunks

Value and computation types

$$A, B \quad ::= \quad \ldots \mid \Sigma x\!:\!A.B \mid \Pi x\!:\!A.B \mid V =_A W \mid U\underline{C} \mid \underline{C} \multimap \underline{D}$$

$$\underline{C}, \underline{D} \quad ::= \quad FA \mid \Sigma x\!:\!A.\underline{C} \mid \Pi x\!:\!A.\underline{C}$$

Value, computation, and homomorphism terms

$$V, W \quad ::= \quad x \mid \ldots \mid \mathtt{thunk}\, M \mid \lambda z\!:\!\underline{C}.K$$

$$\begin{aligned} M, N \quad ::= \quad & \mathtt{force}_{\underline{C}}\, V \mid \mathtt{return}\, V \mid M \;\mathtt{to}\; x\!:\!A \;\mathtt{in}_{\underline{C}}\; N \\ \mid \quad & \langle V, M \rangle \mid M \;\mathtt{to}\; (x\!:\!A, z\!:\!\underline{C}) \;\mathtt{in}_{\underline{D}}\; K \mid \ldots \end{aligned}$$

$$K, L \quad ::= \quad z \mid K \;\mathtt{to}\; x\!:\!A \;\mathtt{in}_{\underline{C}}\; M \mid \langle V, K \rangle \mid \ldots$$

Chapter 4: Fibred adjunction models

A natural combination of

- fibrational models of dependent types
- adjunction-based models of computational effects

as depicted in

$$
\begin{tikzcd}
\mathcal{B}^{\to} \arrow[dr, bend right, "\mathsf{cod}"'] & \mathcal{V} \arrow[l, "\mathcal{P}"', "\mathsf{f+f}"] \arrow[rr, bend left, "F"] \arrow[d, bend right=50, "p"'] \arrow[d, bend left=50, "\{-\}"] \arrow[rr, phantom, "\perp"] & & \mathcal{C} \arrow[ll, bend left, "U"] \arrow[dll, bend left, "q"] \\
& \mathcal{B} \arrow[u, "1"] \arrow[u, phantom, shift right=4, "\dashv"] \arrow[u, phantom, shift left=4, "\dashv"] & &
\end{tikzcd}
$$

Also in Chapter 4

- structures for modelling eMLTT's value and comp. types
- examples of such models, based on models of EEC, families of sets fibration, fibred EM-resolution, and continuous families fibration

Chapter 4: Fibred adjunction models

A natural combination of

- fibrational models of dependent types
- adjunction-based models of computational effects

as depicted in

$$
\begin{tikzcd}
\mathcal{B}^{\to} \arrow[dr, bend right, "\mathsf{cod}"'] & \mathcal{V} \arrow[l, "\mathcal{P}"', "\mathsf{f{+}f}"] \arrow[r, bend left, "F"] \arrow[r, phantom, "\perp"] \arrow[d, bend right=50, "p"'] \arrow[d, bend left=50, "\{-\}"] & \mathcal{C} \arrow[l, bend left, "U"] \arrow[dl, bend left, "q"] \\
& \mathcal{B} \arrow[u, "\dashv", "1 \dashv"'] &
\end{tikzcd}
$$

Also in Chapter 4

- structures for modelling eMLTT's value and comp. types
- examples of such models, based on models of EEC, families of sets fibration, fibred EM-resolution, and continuous families fibration

Chapter 4: Fibred adjunction models

A natural combination of

- fibrational models of dependent types
- adjunction-based models of computational effects

as depicted in

$$
\begin{tikzcd}[column sep=huge, row sep=huge]
\mathcal{B}^{\rightarrow} \arrow[dr, bend right, "\mathsf{cod}"'] & \mathcal{V} \arrow[l, "\mathcal{P}"', "\mathsf{f+f}"] \arrow[d, bend right=60, "p"'] \arrow[d, bend left=60, "\{-\}"] \arrow[rr, bend left, "F"] \arrow[rr, phantom, "\perp"] & & \mathcal{C} \arrow[ll, bend left, "U"] \arrow[dll, bend left, "q"] \\
& \mathcal{B} \arrow[u, "1"' near start, "\dashv" description, "\dashv"] & &
\end{tikzcd}
$$

Also in Chapter 4

- structures for modelling eMLTT's value and comp. types
- examples of such models, based on models of EEC, families of sets fibration, fibred EM-resolution, and continuous families fibration

Chapter 4: Fibred adjunction models

A natural combination of

- fibrational models of dependent types
- adjunction-based models of computational effects

as depicted in

$$
\begin{tikzcd}[column sep=huge, row sep=huge]
\mathcal{B}^{\to} \arrow[dr, bend right, "\mathsf{cod}"'] & \mathcal{V} \arrow[l, "\mathcal{P}"', "\mathsf{f+f}"] \arrow[rr, bend left, "F"] \arrow[rr, phantom, "\perp"] \arrow[d, bend right=50, "p"'] \arrow[d, bend left=50, "\{-\}"] & & \mathcal{C} \arrow[ll, bend left, "U"] \arrow[dll, bend left, "q"] \\
& \mathcal{B} \arrow[u, "\dashv", "1 \dashv"'] & &
\end{tikzcd}
$$

Also in Chapter 4

- structures for modelling eMLTT's value and comp. types
- examples of such models, based on models of EEC, families of sets fibration, fibred EM-resolution, and continuous families fibration

Chapter 5: Interpretation of eMLTT

Interpretation $[\![-]\!]$

- an a priori partial interpretation function that, if defined, maps
  - contexts to objects of $\mathcal{B}$
  - value types to objects of $\mathcal{V}$ and computation types to objects of $\mathcal{C}$
  - value and computation terms to global elements in the fibres of $\mathcal{V}$
  - homomorphism terms to vertical morphisms in $\mathcal{C}$

Soundness

- weakening is interpreted as reindexing along $\pi_{[\![\Gamma; A]\!]}$
- substitution is interpreted as reindexing along $\mathsf{s}([\![\Gamma; V]\!])$
- $[\![-]\!]$ is defined on well-formed syntax
- $[\![-]\!]$ identifies definitionally equal contexts, types, and terms

Completeness

- constructing a classifying model from the syntax of eMLTT

Chapter 5: Interpretation of eMLTT

Interpretation $[\![-]\!]$

- an a priori partial interpretation function that, if defined, maps
  - contexts to objects of $\mathcal{B}$
  - value types to objects of $\mathcal{V}$ and computation types to objects of $\mathcal{C}$
  - value and computation terms to global elements in the fibres of $\mathcal{V}$
  - homomorphism terms to vertical morphisms in $\mathcal{C}$

Soundness

- weakening is interpreted as reindexing along $\pi_{[\![\Gamma;A]\!]}$
- substitution is interpreted as reindexing along $\mathsf{s}([\![\Gamma;V]\!])$
- $[\![-]\!]$ is defined on well-formed syntax
- $[\![-]\!]$ identifies definitionally equal contexts, types, and terms

Completeness

- constructing a classifying model from the syntax of eMLTT

Chapter 5: Interpretation of eMLTT

Interpretation $[\![-]\!]$

- an a priori partial interpretation function that, if defined, maps
  - contexts to objects of $\mathcal{B}$
  - value types to objects of $\mathcal{V}$ and computation types to objects of $\mathcal{C}$
  - value and computation terms to global elements in the fibres of $\mathcal{V}$
  - homomorphism terms to vertical morphisms in $\mathcal{C}$

Soundness

- weakening is interpreted as reindexing along $\pi_{[\![\Gamma;\, A]\!]}$
- substitution is interpreted as reindexing along $\mathsf{s}([\![\Gamma; V]\!])$
- $[\![-]\!]$ is defined on well-formed syntax
- $[\![-]\!]$ identifies definitionally equal contexts, types, and terms

Completeness

- constructing a classifying model from the syntax of eMLTT

Chapter 5: Interpretation of eMLTT

Interpretation $[\![-]\!]$

- an a priori partial interpretation function that, if defined, maps
  - contexts to objects of $\mathcal{B}$
  - value types to objects of $\mathcal{V}$ and computation types to objects of $\mathcal{C}$
  - value and computation terms to global elements in the fibres of $\mathcal{V}$
  - homomorphism terms to vertical morphisms in $\mathcal{C}$

Soundness

- weakening is interpreted as reindexing along $\pi_{[\![\Gamma;\, A]\!]}$
- substitution is interpreted as reindexing along $\mathsf{s}([\![\Gamma; V]\!])$
- $[\![-]\!]$ is defined on well-formed syntax
- $[\![-]\!]$ identifies definitionally equal contexts, types, and terms

Completeness

- constructing a classifying model from the syntax of eMLTT

Chapter 6: eMLTT with algebraic effects

Fibred effect theories $\mathcal{T}_{\textbf{eff}} = (\mathcal{S}_{\textbf{eff}}, \mathcal{E}_{\textbf{eff}})$

- dependently typed operation symbols $\mathsf{op} : (x : I) \longrightarrow O$
- effect terms $T ::= w(V) \mid \mathsf{op}_V(y.T) \mid \ldots$
- equations $\Gamma \mid \Delta \vdash T_1 = T_2$
- examples
  - global state with location-dependent store types
  - dependently typed update monads

Algebraic effects in $\textbf{eMLTT}_{\mathcal{T}_{\textbf{eff}}}$

- algebraic operations as comp. terms $M ::= \ldots \mid \mathsf{op}^{\underline{C}}_V(y.M)$
- equations are included via translation $(\!| \Gamma \mid \Delta \vdash T |\!)_{\underline{A}, \overrightarrow{V_i}, \overrightarrow{V'_j}, \overrightarrow{W_{\mathsf{op}}}}$
- general algebraicity equation (for $\Gamma \mid z : \underline{C} \vDash_{\mathsf{h}} K : \underline{D}$)

$$\Gamma \vDash_{\mathsf{c}} K[\mathsf{op}^{\underline{C}}_V(y.M)/z] = \mathsf{op}^{\underline{D}}_V(y.K[M/z]) : \underline{D}$$

Chapter 6: eMLTT with algebraic effects

Fibred effect theories $\mathcal{T}_{\mathbf{eff}} = (\mathcal{S}_{\mathbf{eff}}, \mathcal{E}_{\mathbf{eff}})$

- dependently typed operation symbols $\mathsf{op} : (x : I) \longrightarrow O$
- effect terms $T ::= w(V) \;\mid\; \mathsf{op}_V(y.T) \;\mid\; \ldots$
- equations $\Gamma \mid \Delta \vdash T_1 = T_2$
- examples
  - global state with location-dependent store types
  - dependently typed update monads

Algebraic effects in eMLTT$_{\mathcal{T}_{\mathbf{eff}}}$

- algebraic operations as comp. terms $M ::= \ldots \;\mid\; \mathsf{op}^{\underline{C}}_V(y.M)$
- equations are included via translation $(\!| \Gamma \mid \Delta \vdash T |\!)_{A, \overrightarrow{V_i}, \overrightarrow{V'_j}, \overrightarrow{W_{\mathsf{op}}}}$
- general algebraicity equation (for $\Gamma \mid z : \underline{C} \vDash_{\mathsf{h}} K : \underline{D}$)

$$\Gamma \vDash_{\mathsf{c}} K[\mathsf{op}^{\underline{C}}_V(y.M)/z] = \mathsf{op}^{\underline{D}}_V(y.K[M/z]) : \underline{D}$$

Chapter 6: eMLTT with algebraic effects

Fibred effect theories $\mathcal{T}_{\mathsf{eff}} = (\mathcal{S}_{\mathsf{eff}}, \mathcal{E}_{\mathsf{eff}})$

- dependently typed operation symbols $\mathsf{op} : (x : I) \longrightarrow O$
- effect terms $T ::= w(V) \ \mid\ \mathsf{op}_V(y.T) \ \mid\ \ldots$
- equations $\Gamma \mid \Delta \vdash T_1 = T_2$
- examples
  - global state with location-dependent store types
  - dependently typed update monads

Algebraic effects in eMLTT$_{\mathcal{T}_{\mathsf{eff}}}$

- algebraic operations as comp. terms $M ::= \ \ldots \ \mid\ \mathsf{op}^{\underline{C}}_V(y.M)$
- equations are included via translation $(\!| \Gamma \mid \Delta \vdash T |\!)_{A, \overrightarrow{V_i}, \overrightarrow{V'_j}, \overrightarrow{W_{\mathsf{op}}}}$
- general algebraicity equation (for $\Gamma \mid z : \underline{C} \vDash_{\mathsf{h}} K : \underline{D}$)

$$\Gamma \vDash_{\mathsf{c}} K[\mathsf{op}^{\underline{C}}_V(y.M)/z] = \mathsf{op}^{\underline{D}}_V(y.K[M/z]) : \underline{D}$$

Chapter 6: eMLTT with algebraic effects

Fibred effect theories $\mathcal{T}_{\textbf{eff}} = (\mathcal{S}_{\textbf{eff}}, \mathcal{E}_{\textbf{eff}})$

- dependently typed operation symbols $\mathsf{op} : (x : I) \longrightarrow O$
- effect terms $T ::= w(V) \;\mid\; \mathsf{op}_V(y.T) \;\mid\; \ldots$
- equations $\Gamma \mid \Delta \vdash T_1 = T_2$
- examples
  - global state with location-dependent store types
  - dependently typed update monads

Algebraic effects in $\text{eMLTT}_{\mathcal{T}_{\textbf{eff}}}$

- algebraic operations as comp. terms $M ::= \;\ldots\; \mid \; \mathsf{op}^{\underline{C}}_V(y.M)$
- equations are included via translation $(\!| \Gamma \mid \Delta \vdash T |\!)_{A,\overrightarrow{V_i},\overrightarrow{V'_j},\overrightarrow{W_{\mathsf{op}}}}$
- general algebraicity equation (for $\Gamma \mid z : \underline{C} \vdash_{\mathsf{h}} K : \underline{D}$)

$$\Gamma \vdash_{\mathsf{c}} K[\mathsf{op}^{\underline{C}}_V(y.M)/z] = \mathsf{op}^{\underline{D}}_V(y.K[M/z]) : \underline{D}$$

Chapter 7: eMLTT$_{\mathcal{T}_{\text{eff}}}$ with handlers

Problem with the conventional term-level def. of handlers

- as handling denotes a homomorphism, natural to include

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

- but then can prove unsound equations such as

$$\Gamma \vDash_{\mathsf{c}} \texttt{write}^{F1}_{\texttt{true}}(\texttt{return}\,\star) = \texttt{write}^{F1}_{\texttt{false}}(\texttt{return}\,\star) : F1$$

Handlers in eMLTT$^{\mathcal{H}}_{\mathcal{T}_{\text{eff}}}$

- user-defined algebra type $\underline{C} ::= \ \ldots \ \mid \ \langle A, \{V_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\text{eff}}} \rangle$
- composition operations $M ::= \ \ldots \ \mid \ M \;\texttt{as}\; x\!:\!U\underline{C} \;\texttt{in}_{\underline{D}}\; N$

$$K ::= \ \ldots \ \mid \ K \;\texttt{as}\; x\!:\!U\underline{C} \;\texttt{in}_{\underline{D}}\; N$$

This extension enables to

- derive the conventional presentation of handlers and handling
- define predicates $\Gamma \vDash_{\mathsf{v}} V : UFA \to \mathsf{VU}$ on effectful computations

Chapter 7: eMLTT$_{\mathcal{T}_{\text{eff}}}$ with handlers

Problem with the conventional term-level def. of handlers

- as handling denotes a homomorphism, natural to include

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

- but then can prove unsound equations such as

$$\Gamma \vdash_{\!c} \texttt{write}^{F1}_{\texttt{true}}(\texttt{return}\,\star) = \texttt{write}^{F1}_{\texttt{false}}(\texttt{return}\,\star) : F1$$

Handlers in eMLTT$^{\mathcal{H}}_{\mathcal{T}_{\text{eff}}}$

- user-defined algebra type $\underline{C} ::= \;\ldots\; \mid\; \langle A, \{V_{\text{op}}\}_{\text{op} \in \mathcal{S}_{\text{eff}}}\rangle$
- composition operations $M ::= \;\ldots\; \mid\; M \;\texttt{as}\; x\!:\!U\underline{C} \;\texttt{in}_{\underline{D}}\; N$

$$K ::= \;\ldots\; \mid\; K \;\texttt{as}\; x\!:\!U\underline{C} \;\texttt{in}_{\underline{D}}\; N$$

This extension enables to

- derive the conventional presentation of handlers and handling
- define predicates $\Gamma \vdash_{\!v} V : UFA \to \mathsf{VU}$ on effectful computations

Chapter 7: $\mathrm{eMLTT}_{\mathcal{T}_{\mathsf{eff}}}$ with handlers

Problem with the conventional term-level def. of handlers

- as handling denotes a homomorphism, natural to include

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

- but then can prove unsound equations such as

$$\Gamma \vDash_{c} \mathtt{write}^{F1}_{\mathtt{true}}(\mathtt{return}\,\star) = \mathtt{write}^{F1}_{\mathtt{false}}(\mathtt{return}\,\star) : F1$$

Handlers in $\mathrm{eMLTT}^{\mathcal{H}}_{\mathcal{T}_{\mathsf{eff}}}$

- user-defined algebra type $\underline{C} ::= \ \ldots \ \mid \ \langle A, \{V_{\mathtt{op}}\}_{\mathtt{op} \in \mathcal{S}_{\mathsf{eff}}} \rangle$
- composition operations $M ::= \ \ldots \ \mid \ M \;\mathtt{as}\; x\!:\!U\underline{C} \;\mathtt{in}_{\underline{D}}\; N$

$$K ::= \ \ldots \ \mid \ K \;\mathtt{as}\; x\!:\!U\underline{C} \;\mathtt{in}_{\underline{D}}\; N$$

This extension enables to

- derive the conventional presentation of handlers and handling
- define predicates $\Gamma \vDash_{v} V : UFA \to \mathrm{VU}$ on effectful computations

Chapter 7: $\text{eMLTT}_{\mathcal{T}_{\text{eff}}}$ with handlers

Problem with the conventional term-level def. of handlers

- as handling denotes a homomorphism, natural to include

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

- but then can prove unsound equations such as

$$\Gamma \vDash_{\mathsf{c}} \texttt{write}^{F1}_{\texttt{true}}(\texttt{return}\,\star) = \texttt{write}^{F1}_{\texttt{false}}(\texttt{return}\,\star) : F1$$

Handlers in $\text{eMLTT}^{\mathcal{H}}_{\mathcal{T}_{\text{eff}}}$

- user-defined algebra type $\underline{C} ::= \ \ldots \ \mid \ \langle A, \{V_{\texttt{op}}\}_{\texttt{op} \in \mathcal{S}_{\text{eff}}} \rangle$
- composition operations $M ::= \ \ldots \ \mid \ M \ \texttt{as} \ x : U\underline{C} \ \texttt{in}_{\underline{D}} \ N$

$$K ::= \ \ldots \ \mid \ K \ \texttt{as} \ x : U\underline{C} \ \texttt{in}_{\underline{D}} \ N$$

This extension enables to

- derive the conventional presentation of handlers and handling
- define predicates $\Gamma \vDash_{\mathsf{v}} V : UFA \to \mathsf{VU}$ on effectful computations

Thesis structure

- **Chapter 1:** Introduction and related work
- **Chapter 2:** Preliminaries of models of effects and dep. types
- **Chapter 3:** The core language eMLTT
- **Chapter 4:** Categorical models of eMLTT
- **Chapter 5:** Interpretation of eMLTT, soundness, completeness
- **Chapter 6:** An extension of eMLTT with algebraic effects
- **Chapter 7:** An extension of eMLTT with handlers
- **Chapter 8:** Conclusion and future work
- **Appendices:** Details of longer proofs