Fibred Computational Effects

(thesis overview)

Danel Ahman

June 29, 2017

Overall aims

Investigate language design principles for combining

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

The goal of this work was to establish that

- dep. types and comp. effects admit a natural combination
- the 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

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

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

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

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

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

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

Value, computation, and homomorphism terms

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

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

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

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

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

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

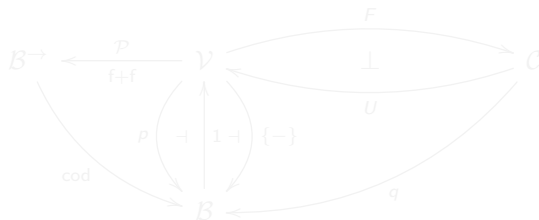**Value, computation, and homomorphism terms**

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

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

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

Chapter 4: Fibred adjunction models

A natural combination of

- fibrational models of dependent types
- adjunction-based models of computational effects
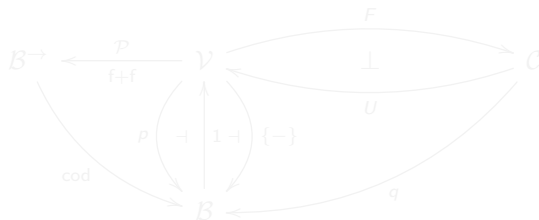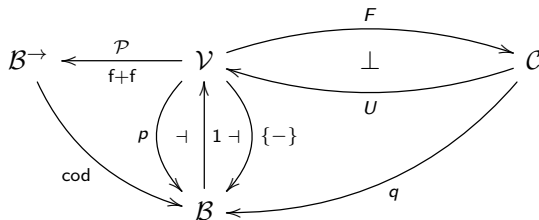
as depicted in

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

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

Chapter 4: Fibred adjunction models

A natural combination of

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

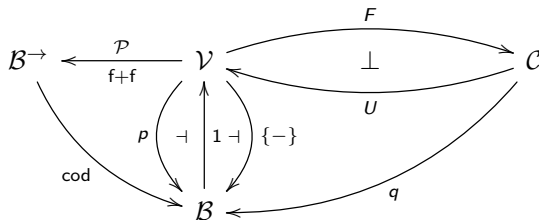**as depicted in**

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 $s([\![\Gamma; V]\!])$
- $[\![-]\!]$ 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 $s([\![\Gamma;V]\!])$
- $[\![-]\!]$ 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 $\mathrm{s}([\![\Gamma;V]\!])$
- $[\![-]\!]$ 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 $s([\![\Gamma;V]\!])$
- $[\![-]\!]$ 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 $\mathcal{T}_{\mathsf{eff}} = (\mathcal{S}_{\mathsf{eff}}, \mathcal{E}_{\mathsf{eff}})$

- 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 $\mathsf{eMLTT}_{\mathcal{T}_{\mathsf{eff}}}$

- algebraic operations as comp. terms $M ::= \ldots \mid \mathsf{op}_V^{\underline{C}}(y.M)$
- 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 $\Gamma \mid z\!:\!\underline{C} \vdash_{\!\!\mathsf{k}} K : \underline{D}$)

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

Chapter 6: eMLTT with algebraic effects

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

- dependently typed operation symbols $\text{op} : (x \!:\! I) \longrightarrow O$
- effect terms $T ::= w(V) \mid \text{op}_V(y.T) \mid \ldots$
- 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$_{\mathcal{T}_{\text{eff}}}$

- algebraic operations as comp. terms $M ::= \ldots \mid \text{op}_V^C(y.M)$
- equations are included via translation $(\!|\, \Gamma \mid \Delta \vdash T \,|\!)_{A, \overrightarrow{V_i}, \overrightarrow{V_j}, \overrightarrow{W_{op}}}$
- general algebraicity equation (for $\Gamma \mid z \!:\! \underline{C} \vDash K : \underline{D}$)

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

Chapter 6: eMLTT with algebraic effects

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

- dependently typed operation symbols $\text{op} : (x : I) \longrightarrow O$
- effect terms $T ::= w(V) \mid \text{op}_V(y.T) \mid \ldots$
- 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$_{\mathcal{T}_{\text{eff}}}$

- algebraic operations as comp. terms $M ::= \ldots \mid \text{op}_V^{\underline{C}}(y.M)$
- equations are included via translation $(\!|\, \Gamma \mid \Delta \vdash T \,|\!)_{A, \overrightarrow{V_i}, \overrightarrow{V_j}, \overrightarrow{W_{op}}}$
- general algebraicity equation (for $\Gamma \mid z : \underline{C} \vdash_{\kappa} K : \underline{D}$)

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

Chapter 6: eMLTT with algebraic effects

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

- dependently typed operation symbols $\text{op} : (x : I) \longrightarrow O$
- effect terms $T ::= w(V) \mid \text{op}_V(y.T) \mid \ldots$
- 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$_{\mathcal{T}_{\text{eff}}}$

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

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

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

Problem with the conventional term-level def. of handlers

- as handling denotes a homomorphism, natural to include

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

- but then can prove unsound equations such as

    $\Gamma \vDash \text{write}_{\text{true}}^{F1}(\text{return}\,\star) = \text{write}_{\text{false}}^{F1}(\text{return}\,\star) : F1$

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

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

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

This extension enables to

- derive the conventional presentation of handlers and handling
- define predicates $\Gamma \vDash V : UFA \to VU$ 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 handled with $\{\text{op}_x(x') \mapsto N_{\text{op}}\}_{\text{op} \in \mathcal{S}_{\text{eff}}}$ to $y : A$ in N_{ret}

- but then can prove unsound equations such as

  $\Gamma \vDash_{\overline{\varepsilon}} \text{write}^{F1}_{\text{true}}(\text{return} \star) = \text{write}^{F1}_{\text{false}}(\text{return} \star) : F1$

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

- user-defined algebra type $\underline{C} ::= \dots \mid \langle A, \{V_{\text{op}}\}_{\text{op} \in \mathcal{S}_{\text{eff}}} \rangle$

- composition operations $M ::= \dots \mid M$ as $x : U\underline{C}$ in$_{\underline{D}} N$

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

This extension enables to

- derive the conventional presentation of handlers and handling

- define predicates $\Gamma \vdash V : UFA \to \text{VU}$ on effectful computations

Chapter 7: 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 handled with $\{\mathsf{op}_x(x') \mapsto N_{\mathsf{op}}\}_{\mathsf{op} \in \mathcal{S}_{\mathsf{eff}}}$ to $y\!:\!A$ in N_{ret}

- but then can prove unsound equations such as

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

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

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

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

This extension enables to

- derive the conventional presentation of handlers and handling
- define predicates $\Gamma \vDash_{\overline{\mathsf{v}}} V : UFA \to \mathsf{VU}$ 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 handled with $\{\text{op}_x(x') \mapsto N_{\text{op}}\}_{\text{op} \in \mathcal{S}_{\text{eff}}}$ to $y : A$ in N_{ret}

- but then can prove unsound equations such as

  $\Gamma \vDash_{\overline{\varepsilon}} \text{write}_{\text{true}}^{F1}(\text{return} \star) = \text{write}_{\text{false}}^{F1}(\text{return} \star) : F1$

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

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

  $\qquad\qquad\qquad\qquad\quad K ::= \dots \mid K$ as $x : U\underline{C}$ in$_{\underline{D}}$ N

This extension enables to

- derive the conventional presentation of handlers and handling
- define predicates $\Gamma \vDash_{\overline{\text{v}}} V : UFA \to \text{VU}$ 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