Modular Monad Transformers

Mauro Jaskelioff

Functional Programming Laboratory
School of Computer Science
University of Nottingham

ESOP 2009, University of York, UK

In this talk

- Monadic semantics and modularity.
- The monad transformers approach.
- Well-behaved operations and their lifting.
- General lifting of operations.

Monads and Computational Effects

- Monads model computational effects.
- Monads structure functional programs.
- Basic monadic approach:
  - Distinguish computations and values.
  - Explain a language in terms of an abstract computation type.
  - Explain the computation types.

Effectful Operations

- Computational monads come equipped with operations.
- Examples:
  - State Monad: *get* and *put*.
  - Exception monad: *throw* and *handle*.
  - Continuation monad: *callcc* and *abort*.

Effectful Operations

- Computational monads come equipped with operations.
- Examples:
  - State Monad: *get* and *put*.
  - Exception monad: *throw* and *handle*.
  - Continuation monad: *callcc* and *abort*.
- (Slightly refined) monadic approach:
  - Distinguish computations and values.
  - Explain a language in terms of an abstract computation type **and effect-manipulating operations**.
  - Explain the computation types **and the operations**.

Modular Approaches

- More complex monads $\Rightarrow$ more complex semantics.
- How to construct complex monads modularly?
  - Distributive laws.
  - Coproducts of monads.
  - Monad transformers.
  - Combination of algebraic theories.
- Monad transformers construct monads incrementally.
- Are easily implementable (ideal for DSL).

Lifting Problem

- A monad which supports state and exceptions should support *get*, *put*, *throw*, and *handle*.
- Let's extend the exception monad with the state monad transformer.
- What are the corresponding *throw* and *handle* operations for the combined monad?
- In general: How to lift operations of the underlying monad to the transformed monad?
- Liang et al.(1995) non-modular workaround:

  "Provide ad-hoc liftings of each operation through each monad transformer"

Goal

Identify classes of operations and classes of monad transformers with a uniform lifting

System $F\omega$

- We will work in system $F\omega$.
  (Higher-order polymorphic lambda-calculus)

$$\begin{array}{rr} \text{kinds} & k ::= * \mid k \to k \\ \text{type constuctors} & U ::= X \mid U \to U \mid \forall X : k.\ U \mid \Lambda X : k.\ U \mid U\,U \\ \text{terms} & e ::= x \mid \lambda x : X.\ e \mid e\ e \mid \Lambda X : k.\ e \mid e\,U \end{array}$$

- Can be considered as:
  - A meta-language for a programming language.
  - A programming language in which we embed a DSL.

- We will *express* categorical constructions in $F\omega$.

Functors and Natural Transformations

Definition (Expressible functor)

A pair $\hat{F} = (F, map^F)$ of

- a type constructor $F : * \to *$,
- a term $map^F : \forall X, Y : *. (X \to Y) \to FX \to FY$ such that map^F respects identities and composition.

Functors and Natural Transformations

Definition (Expressible functor)

A pair $\hat{F} = (F, map^F)$ of

- a type constructor $F : * \to *$,
- a term $map^F : \forall X, Y : *. (X \to Y) \to FX \to FY$ such that map^F respects identities and composition.

Definition (Expressible natural transformation from $\hat{F}$ to $\hat{G}$)

Terms $\tau : \hat{F} \dot{\to} \hat{G} = \forall X : *. FX \to GX$ such that

$$map^G_{A,B}\ f \cdot \tau_A = \tau_B \cdot map^F_{A,B}\ f$$

Monads and Monad Morphisms

Definition (Expressible monad)

A triple $\hat{M} = (M, ret^M, bind^M)$ of

- a type constructor $M : * \longrightarrow *$,
- a term $ret^M : \forall X : *. X \longrightarrow MX$,
- a term $bind^M : \forall X, Y : *. MX \longrightarrow (X \longrightarrow MY) \longrightarrow MY$

such that some coherence conditions hold.

Monads and Monad Morphisms

Definition (Expressible monad)

A triple $\hat{M} = (M, ret^M, bind^M)$ of

- a type constructor $M : * \to *$,
- a term $ret^M : \forall X : *. X \to MX$,
- a term $bind^M : \forall X, Y : *. MX \to (X \to MY) \to MY$

such that some coherence conditions hold.

Examples,

- State monad $\mathbf{S}X = S \to (X, S)$.
- Exception monad $\mathbf{X}X = X + Z$.
- Continuation monad $\mathbf{C}X = (X \to R) \to R$.

Monads and Monad Morphisms

Definition (Expressible monad)

A triple $\hat{M} = (M, ret^M, bind^M)$ of

- a type constructor $M : * \to *$,
- a term $ret^M : \forall X : *. X \to MX$,
- a term $bind^M : \forall X, Y : *. MX \to (X \to MY) \to MY$

such that some coherence conditions hold.

Definition (Expressible monad morphism from $\hat{M}$ to $\hat{N}$)

A terms $\xi : \forall X. MX \to NX$ such that the monad operations are respected.

Monad Transformers

Definition (Expressible monad transformer)

A tuple $\hat{T} = (T, ret^T, bind^T, lift^T)$ of

- a type constructor $T : (* \to *) \to (* \to *)$ and terms,
- $ret^T : \hat{M} \to \forall X : *. X \to TMX$,
- $bind^T : \hat{M} \to \forall X, Y : *. TMX \to (X \to TMY) \to TMY$
- $lift^T : \hat{M} \dot{\to} T\hat{M}$

such that for every monad $\hat{M}$, the triple $(TM, ret^T_{\hat{M}}, bind^T_{\hat{M}})$ is a monad and $lift^T_{\hat{M}}$ is a monad morphism.

Monad Transformers

Definition (Expressible monad transformer)

A tuple $\hat{T} = (T, ret^T, bind^T, lift^T)$ of

- a type constructor $T : (* \to *) \to (* \to *)$ and terms,
- $ret^T : \hat{M} \to \forall X : *.\, X \to TMX$,
- $bind^T : \hat{M} \to \forall X, Y : *.\, TMX \to (X \to TMY) \to TMY$
- $lift^T : \hat{M} \dot{\to} T\hat{M}$

such that for every monad $\hat{M}$, the triple $(TM, ret^T_{\hat{M}}, bind^T_{\hat{M}})$ is a monad and $lift^T_{\hat{M}}$ is a monad morphism.

Examples,

- State monad transformer $\mathcal{S}(M)X = S \to M(X, S)$.
- Exception monad transformer $\mathcal{X}(M)X = M(X + Z)$.
- Continuation monad transf. $\mathcal{C}(M)X = (X \to MR) \to MR$.

Operations and Lifting

Definition (Σ-operation for a monad $\hat{M}$)

A natural transformation $op : \Sigma \circ M \dot{\to} M$, where $\hat{\Sigma}$ is a functor.

Operations and Lifting

Definition (Σ-operation for a monad $\hat{M}$)

A natural transformation $op : \Sigma \circ M \dot{\to} M$, where $\hat{\Sigma}$ is a functor.

Examples:

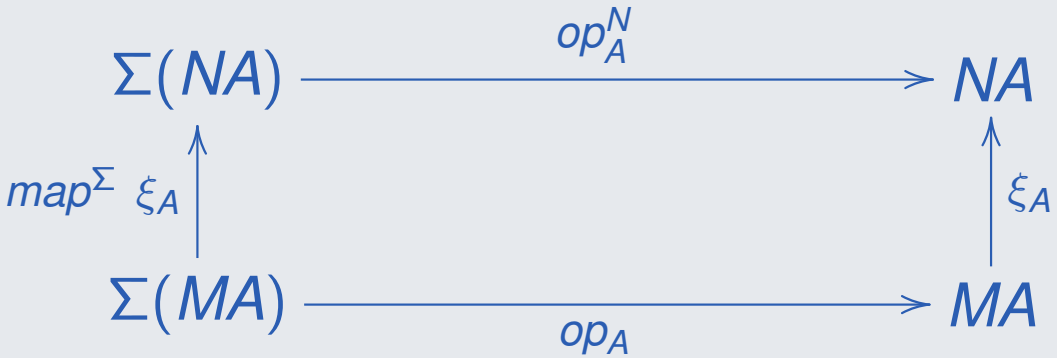
- $get_A : (S \to \mathbf{S}A) \to \mathbf{S}A$ $\qquad \Sigma_g X = (S \to X)$
  $set_A : (S, \mathbf{S}A) \to \mathbf{S}A$ $\qquad \Sigma_s X = S \times X$
- $throw_A : 1 \to \mathbf{X}A$ $\qquad \Sigma_t X = 1$
  $handle_A : \mathbf{X}A \to (Z \to \mathbf{X}A) \to \mathbf{X}A$ $\qquad \Sigma_h X = X \times (Z \to X)$
- $callcc_A : ((\mathbf{C}A \to R) \to \mathbf{C}A) \to \mathbf{C}A$ $\qquad \Sigma_c X = (X \to R) \to X$
  $abort_A : R \to \mathbf{C}A$ $\qquad \Sigma_a X = R$

Operations and Lifting

Definition (Σ-operation for a monad $\hat{M}$)

A natural transformation $op : \Sigma \circ M \dot{\to} M$, where $\hat{\Sigma}$ is a functor.

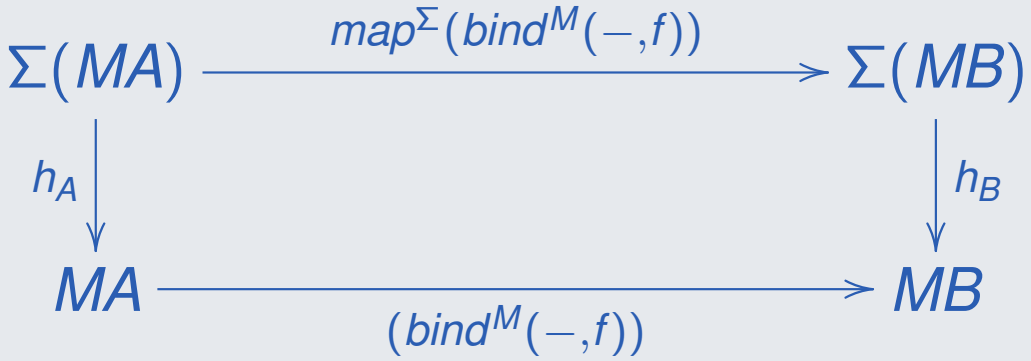
Definition (Lifting along a monad morphism $\xi : \hat{M} \dot{\to} \hat{N}$)



Well-behaved Operations

Definition (Σ-algebraic operation for a monad $\hat{M}$)

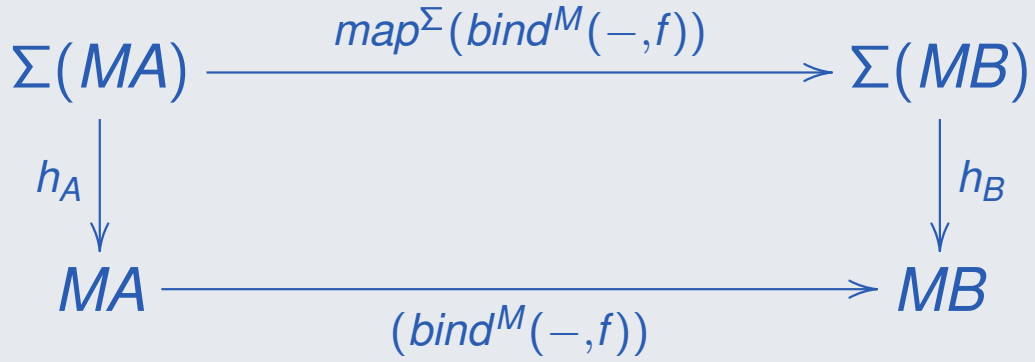
A Σ-operation for $\hat{M}$ such that for any $f : A \to MB$:



Well-behaved Operations

Definition (Σ-algebraic operation for a monad $\hat{M}$)

A Σ-operation for $\hat{M}$ such that for any $f : A \to MB$:



Examples:

- *get* and *set*
- *throw*
- *callcc* and *abort*

Non-example:

- *handle*

Lifting of algebraic operations

Proposition

Σ-*algebraic* $\Sigma \circ M \overset{\bullet}{\to} M \quad \cong \quad \Sigma \overset{\bullet}{\to} M$.

Theorem (Unique Algebraic Lifting)

Σ-*algebraic operations have a unique lifting through a monad morphism* $\xi : \hat{M} \overset{\bullet}{\to} \hat{N}$.

Lifting of algebraic operations

Proposition

Σ-*algebraic* $\Sigma \circ M \overset{\bullet}{\to} M \quad \cong \quad \Sigma \overset{\bullet}{\to} M$.

Theorem (Unique Algebraic Lifting)

Σ-*algebraic operations have a unique lifting through a monad morphism* $\xi : \hat{M} \overset{\bullet}{\to} \hat{N}$.

$\Sigma \circ M \overset{\bullet}{\to} M$ algebraic

Lifting of algebraic operations

Proposition

Σ-*algebraic* $\Sigma \circ M \overset{\bullet}{\to} M \quad \cong \quad \Sigma \overset{\bullet}{\to} M$.

Theorem (Unique Algebraic Lifting)

Σ-*algebraic operations have a unique lifting through a monad morphism* $\xi : \hat{M} \overset{\bullet}{\to} \hat{N}$.

$$\Sigma \overset{\bullet}{\to} M$$

Lifting of algebraic operations

Proposition

Σ-algebraic $\Sigma \circ M \overset{\bullet}{\to} M \quad \cong \quad \Sigma \overset{\bullet}{\to} M$.

Theorem (Unique Algebraic Lifting)

Σ-algebraic operations have a unique lifting through a monad morphism $\xi : \hat{M} \overset{\bullet}{\to} \hat{N}$.

$$\Sigma \overset{\bullet}{\to} M \overset{\xi}{\to} N$$

Lifting of algebraic operations

Proposition

Σ-*algebraic* $\Sigma \circ M \overset{\bullet}{\to} M \quad \cong \quad \Sigma \overset{\bullet}{\to} M$.

Theorem (Unique Algebraic Lifting)

Σ-*algebraic operations have a unique lifting through a monad morphism* $\xi : \hat{M} \overset{\bullet}{\to} \hat{N}$.

$\Sigma \circ N \overset{\bullet}{\to} N$ algebraic

Functorial Monad Transformers

Definition (Functorial monad transformer)

- A monad transformer $(T, ret^T, bind^T, lift^T)$
- A map $hmap^T : \forall \hat{M}, \hat{N}. (M \dot{\to} N) \to (TM \dot{\to} TN)$
- Additional coherence conditions
  - $hmap^T$ should preserve natural transformations and monad morphisms,
  - respect identities and composition of natural transformations,
  - $lift^T$ should be natural (behave well wrt $hmap^T$).

Functorial Monad Transformers

Definition (Functorial monad transformer)

- A monad transformer $(T, ret^T, bind^T, lift^T)$
- A map $hmap^T : \forall \hat{M}, \hat{N}. (M \dot{\to} N) \to (TM \dot{\to} TN)$
- Additional coherence conditions
  - $hmap^T$ should preserve natural transformations and monad morphisms,
  - respect identities and composition of natural transformations,
  - $lift^T$ should be natural (behave well wrt $hmap^T$).

Examples:

- State monad transformer
- Exception monad transformer

Non-example:

- Continuation monad transformer

Codensity Monad Transformer

- $\mathcal{K}MX \mathrel{\hat{=}} \forall Y : *. (X \to MY) \to MY$ is part of a monad transformer.
- Exploits impredicativity of system $F\omega$.
- Properties of $\hat{\mathcal{K}}$:
  - Every Σ-operation $op : \Sigma \circ M \dot{\to} M$ induces a Σ-algebraic operation $op^{\mathcal{K}} : \Sigma \circ \mathcal{K}M \dot{\to} \mathcal{K}M$.
  - $from : \mathcal{K}M \dot{\to} M$, such that $from \cdot lift^{\mathcal{K}} = id$.
  - $op = from \cdot op^{\mathcal{K}} \cdot map^{\Sigma}\ lift^{\mathcal{K}}$.

General Lifting

Theorem (Lifting)

Given

- $op : \Sigma \circ M \bullet\!\!\to M$
- *Functorial monad transformer* $\hat{T}$.

There is a lifting $op^T : \Sigma \circ TM \bullet\!\!\to TM$ *of* op *along* $lift^T$.

$$op^T = \Sigma \circ TM \xrightarrow{\Sigma \circ T\, lift^{\mathcal{K}}} \Sigma \circ T(\mathcal{K}M) \xrightarrow{op^{\mathcal{K},T}} T(\mathcal{K}M) \xrightarrow{from} TM$$

$$\Sigma \circ T\, lift^{\mathcal{K}} = map^{\Sigma}(hmap^{T}(lift^{\mathcal{K}}))$$

General Lifting

Theorem (Lifting)

Given

- $op : \Sigma \circ M \dashrightarrow M$
- *Functorial monad transformer* $\hat{T}$.

There is a lifting $op^T : \Sigma \circ TM \dashrightarrow TM$ *of* op *along* $lift^T$.

$$op^T = \Sigma \circ TM \xrightarrow{\Sigma \circ T\, lift^{\mathcal{K}}} \Sigma \circ T(\mathcal{K}M) \xrightarrow{op^{\mathcal{K},T}} T(\mathcal{K}M) \xrightarrow{from} TM$$

$$\Sigma \circ T\, lift^{\mathcal{K}} = map^{\Sigma}(hmap^{T}(lift^{\mathcal{K}}))$$

- If op is Σ-algebraic, the algebraic lifting and the general lifting coincide.

Example of lifted operations

- For a concrete functorial monad transformer, the lifting simplifies to:
- $\mathcal{S}MX = S \to M(X \times S)$

$$op_X^{\mathcal{S}}\,(t : \Sigma(\mathcal{S}MX)) : \mathcal{S}MX = \lambda s.\, op_{X\times S}(map^{\Sigma}\,\tau^{s}\,t)$$

where $\tau^{s}(f : S \to M(X \times S)) = f\,s$.

Example of lifted operations

- For a concrete functorial monad transformer, the lifting simplifies to:
- $\mathcal{S}MX = S \to M(X \times S)$

$$op_X^{\mathcal{S}}(t : \Sigma(\mathcal{S}MX)) : \mathcal{S}MX = \lambda s.\, op_{X \times S}(map^{\Sigma}\, \tau^s\, t)$$

where $\tau^s(f : S \to M(X \times S)) = f\, s$.

- $\mathcal{X}MX = M(Z + X)$

$$op_X^{\mathcal{X}}(t : \Sigma(\mathcal{X}MX)) : \mathcal{X}MX = op_{Z+X}\, t.$$

Summary

- Uniform liftings for a wide class of operations and monad transformers.
- Expressible categorical constructs essential for finding a solution to the problem.
- Future Work.
  - Extend to “mixed-variant” transformers.
  - Give a purely abstract account.
  - General lifting without relying on impredicativity.
  - Extend these results beyond monads.