Monads and the Curry-Howard Isomorphism

Christoph Rauch

FAU Erlangen-Nürnberg
Theorieseminar WS2012

December 17, 2012

$$\frac{\Gamma \vdash e : A}{\Gamma \vdash \mathrm{ret}\ e : TA} \qquad \frac{\Gamma, x : A \vdash f : TB}{\Gamma \vdash \mathrm{do}\ x \leftarrow \mathrm{ret}\ e;\ f : TB}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash \bigcirc A} \qquad \frac{\Gamma, A \vdash \bigcirc B}{\Gamma \vdash \bigcirc B}$$

Intuitionistic Propositional Logic

Why care?

Intuitionistic Propositional Logic

Why care?

- *Constructive* logic

Intuitionistic Propositional Logic

Why care?

- *Constructive* logic
- *Tertium non datur* rejection

Intuitionistic Propositional Logic

Why care?

- *Constructive* logic
- *Tertium non datur* rejection

Semantics

- A proof of $\varphi_1 \wedge \varphi_2$ consists of a proof of φ_1 and a proof of φ_2
- A proof of $\varphi_1 \vee \varphi_2$ consists of a number $i \in \{1, 2\}$ and a proof of φ_i
- A proof of $\varphi_1 \to \varphi_2$ is a method transforming a proof of φ_1 into a proof of φ_2
- There is no proof of $\bot$

Natural Deduction rules

$$\frac{}{\top}\ \top\mathsf{I} \qquad \frac{\bot}{C}\ \bot\mathsf{E}$$

$$\frac{A \quad B}{A \wedge B}\ \wedge\mathsf{I} \qquad \frac{A \wedge B}{A}\ \wedge\mathsf{E}^{\mathsf{L}} \qquad \frac{A \wedge B}{B}\ \wedge\mathsf{E}^{\mathsf{R}}$$

$$\frac{\begin{array}{c}\overline{A}\ a \\ \vdots \\ B\end{array}}{A \to B}\ \to\mathsf{I}_a \qquad \frac{A \to B \quad A}{B}\ \to\mathsf{E}$$

$$\frac{A}{A \vee B}\ \vee\mathsf{I}^{\mathsf{L}} \qquad \frac{B}{A \vee B}\ \vee\mathsf{I}^{\mathsf{R}}$$

$$\frac{A \vee B \quad \begin{array}{c}\overline{A}\ a \\ \vdots \\ C\end{array} \quad \begin{array}{c}\overline{B}\ b \\ \vdots \\ C\end{array}}{C}\ \vee\mathsf{E}_{a,b}$$

Deducing the Lambda Calculus

What next?

- Rule system above describes proof trees locally
- $\Rightarrow$ Develop a *meta*-system for proof objects
- Ternary symbol $\Gamma \vdash \mathcal{P} : P$ for proof trees $\mathcal{P}$ of a proposition P under hypotheses Γ

Simply Typed Lambda Calculus with Products & Coproducts

$$\frac{}{\Gamma \vdash * : \top}\ \lambda\top\mathsf{I} \qquad \frac{}{\Gamma, p : P \vdash p : P}\ \lambda\mathsf{hyp}$$

$$\frac{\Gamma \vdash \mathcal{A} : A \qquad \Gamma \vdash \mathcal{B} : B}{\Gamma \vdash \langle \mathcal{A}, \mathcal{B} \rangle : A \wedge B}\ \lambda\wedge\mathsf{I} \qquad \frac{\Gamma \vdash \mathcal{L} : \bot}{\Gamma \vdash \bot\mathsf{elim}\ \mathcal{L} : C}\ \lambda\bot\mathsf{E}$$

$$\frac{\Gamma, a : A \vdash \mathcal{B} : B}{\Gamma \vdash \lambda a.\mathcal{B} : A \to B}\ \lambda\to\mathsf{I} \qquad \frac{\Gamma \vdash \mathcal{P} : A \wedge B}{\Gamma \vdash \mathsf{fst}\ \mathcal{P} : A}\ \lambda\wedge\mathsf{E}^{\mathsf{L}}$$

$$\frac{\Gamma \vdash \mathcal{A} : A}{\Gamma \vdash \mathsf{inl}\ \mathcal{A} : A \vee B}\ \lambda\vee\mathsf{I}^{\mathsf{L}} \qquad \frac{\Gamma \vdash \mathcal{P} : A \wedge B}{\Gamma \vdash \mathsf{snd}\ \mathcal{P} : B}\ \lambda\wedge\mathsf{E}^{\mathsf{R}}$$

$$\frac{\Gamma \vdash \mathcal{B} : B}{\Gamma \vdash \mathsf{inr}\ \mathcal{B} : A \vee B}\ \lambda\vee\mathsf{I}^{\mathsf{R}} \qquad \frac{\Gamma \vdash \mathcal{F} : A \to B \qquad \Gamma \vdash \mathcal{A} : A}{\Gamma \vdash \mathcal{F}\mathcal{A} : B}\ \lambda\to\mathsf{E}$$

$$\frac{\Gamma \vdash \mathcal{D} : A \vee B \qquad \Gamma, a : A \vdash \mathcal{C} : C \qquad \Gamma, b : B \vdash \mathcal{C}' : C}{\Gamma \vdash \mathsf{case}\ \mathcal{D}\ \mathsf{of}\ \{\ \mathsf{inl}\ a \mapsto \mathcal{C}\ ;\ \mathsf{inr}\ b \mapsto \mathcal{C}'\ \} : C}\ \lambda\vee\mathsf{E}$$

Example λ Term

$\lambda f.\lambda g.\lambda a.f\,(g\ a)$

Let $\Gamma = \{\ f\ :\ B \to C,\ g\ :\ A \to B,\ a\ :\ A\ \}$

Example λ Term

$\lambda f.\lambda g.\lambda a.f\,(g\;a)$

Let $\Gamma = \{\; f \;:\; B \to C,\; g \;:\; A \to B,\; a \;:\; A \;\}$

$$\frac{}{\Gamma \vdash f : B \to C}\;\lambda\text{hyp} \qquad \frac{}{\Gamma \vdash g : A \to B}\;\lambda\text{hyp} \qquad \frac{}{\Gamma \vdash a : A}\;\lambda\text{hyp}$$

Example λ Term

$\lambda f.\lambda g.\lambda a.f\,(g\ a)$

Let $\Gamma = \{\ f\ :\ B \to C,\ g\ :\ A \to B,\ a\ :\ A\ \}$

$$\dfrac{}{\Gamma \vdash f : B \to C}\ \lambda\text{hyp} \qquad \dfrac{\dfrac{}{\Gamma \vdash g : A \to B}\ \lambda\text{hyp} \qquad \dfrac{}{\Gamma \vdash a : A}\ \lambda\text{hyp}}{\Gamma \vdash g\ a : B}\ \lambda{\to}\text{E}$$

Example λ Term

$\lambda f.\lambda g.\lambda a.f\,(g\ a)$

Let $\Gamma = \{\ f\ :\ B \to C,\ g\ :\ A \to B,\ a\ :\ A\ \}$

$$\dfrac{\dfrac{}{\Gamma \vdash f : B \to C}\,\lambda\text{hyp} \qquad \dfrac{\dfrac{}{\Gamma \vdash g : A \to B}\,\lambda\text{hyp} \qquad \dfrac{}{\Gamma \vdash a : A}\,\lambda\text{hyp}}{\Gamma \vdash g\ a : B}\,\lambda{\to}\text{E}}{\Gamma \vdash f\,(g\ a) : C}\,\lambda{\to}\text{E}$$

Example λ Term

$\lambda f.\lambda g.\lambda a.f\,(g\ a)$

Let $\Gamma = \{\ f\ :\ B \to C,\ g\ :\ A \to B,\ a\ :\ A\ \}$

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash f : B \to C}\,\lambda\text{hyp} \qquad \dfrac{\dfrac{}{\Gamma \vdash g : A \to B}\,\lambda\text{hyp} \qquad \dfrac{}{\Gamma \vdash a : A}\,\lambda\text{hyp}}{\Gamma \vdash g\ a : B}\,\lambda{\to}\text{E}}{\Gamma \vdash f\,(g\ a) : C}\,\lambda{\to}\text{E}}{f : B \to C, g : A \to B \vdash \lambda a.f\,(g\ a) : A \to C}\,\lambda{\to}\text{I}$$

Example λ Term

$\lambda f.\lambda g.\lambda a.f\,(g\ a)$

Let $\Gamma = \{\ f\ :\ B \to C,\ g\ :\ A \to B,\ a\ :\ A\ \}$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash f : B \to C}\,\lambda\text{hyp} \qquad \dfrac{\dfrac{}{\Gamma \vdash g : A \to B}\,\lambda\text{hyp} \qquad \dfrac{}{\Gamma \vdash a : A}\,\lambda\text{hyp}}{\Gamma \vdash g\ a : B}\,\lambda{\to}\text{E}}{\Gamma \vdash f\,(g\ a) : C}\,\lambda{\to}\text{E}}{f : B \to C, g : A \to B \vdash \lambda a.f\,(g\ a) : A \to C}\,\lambda{\to}\text{I}}{f : B \to C \vdash \lambda g.\lambda a.f\,(g\ a) : (A \to B) \to A \to C}\,\lambda{\to}\text{I}$$

Example λ Term

$\lambda f.\lambda g.\lambda a.f\,(g\ a)$

Let $\Gamma = \{\ f\ :\ B \to C,\ g\ :\ A \to B,\ a\ :\ A\ \}$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash f : B \to C}\,\lambda\text{hyp} \quad \dfrac{\dfrac{}{\Gamma \vdash g : A \to B}\,\lambda\text{hyp} \quad \dfrac{}{\Gamma \vdash a : A}\,\lambda\text{hyp}}{\Gamma \vdash g\ a : B}\,\lambda{\to}\text{E}}{\Gamma \vdash f\,(g\ a) : C}\,\lambda{\to}\text{E}}{f : B \to C, g : A \to B \vdash \lambda a.f\,(g\ a) : A \to C}\,\lambda{\to}\text{I}}{f : B \to C \vdash \lambda g.\lambda a.f\,(g\ a) : (A \to B) \to A \to C}\,\lambda{\to}\text{I}}{\vdash \lambda f.\lambda g.\lambda a.f\,(g\ a) : (B \to C) \to (A \to B) \to A \to C}\,\lambda{\to}\text{I}$$

Example λ term

$\lambda f.\lambda g.\lambda a.f\,(g\ a)$

Corresponding IPL deduction:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{B \to C}\,f \quad \dfrac{\dfrac{}{A \to B}\,g \quad \dfrac{}{A}\,a}{B}\,\to\mathsf{E}}{C}\,\to\mathsf{E}}{A \to C}\,\to\mathsf{I}_a}{(A \to B) \to A \to C}\,\to\mathsf{I}_g}{(B \to C) \to (A \to B) \to A \to C}\,\to\mathsf{I}_f$$

Properties

- Equational theory of $\beta\eta$-equivalence yields soundness/local completeness on logical side

Properties

- Equational theory of $\beta\eta$-equivalence yields soundness/local completeness on logical side

e.g. $$\Gamma \vdash (\lambda a.\mathcal{B})\ \mathcal{A} \ :\ B \quad \rightsquigarrow \quad \Gamma \vdash \mathcal{B}[\mathcal{A}/a] \ :\ B$$

Properties

- Equational theory of $\beta\eta$-equivalence yields soundness/local completeness on logical side
- *Not* Turing-complete – every program terminates

Properties

- Equational theory of $\beta\eta$-equivalence yields soundness/local completeness on logical side
- *Not* Turing-complete – every program terminates
- Naturally extendible by adding more rules (while staying *simply typed*)

Properties

- Equational theory of $\beta\eta$-equivalence yields soundness/local completeness on logical side
- *Not* Turing-complete – every program terminates
- Naturally extendible by adding more rules (while staying *simply typed*)

What is impossible?

Properties

- Equational theory of $\beta\eta$-equivalence yields soundness/local completeness on logical side
- *Not* Turing-complete – every program terminates
- Naturally extendible by adding more rules (while staying *simply typed*)

What is impossible?

- Exceptions

Properties

- Equational theory of $\beta\eta$-equivalence yields soundness/local completeness on logical side
- *Not* Turing-complete – every program terminates
- Naturally extendible by adding more rules (while staying *simply typed*)

What is impossible?

- Exceptions
- Input/Output

Properties

- Equational theory of $\beta\eta$-equivalence yields soundness/local completeness on logical side
- *Not* Turing-complete – every program terminates
- Naturally extendible by adding more rules (while staying *simply typed*)

What is impossible?

- Exceptions
- Input/Output
- State

Properties

- Equational theory of $\beta\eta$-equivalence yields soundness/local completeness on logical side
- *Not* Turing-complete – every program terminates
- Naturally extendible by adding more rules (while staying *simply typed*)

What is impossible?

- Exceptions
- Input/Output
- State
- Generally: side effects

Properties

- Equational theory of $\beta\eta$-equivalence yields soundness/local completeness on logical side
- *Not* Turing-complete – every program terminates
- Naturally extendible by adding more rules (while staying *simply typed*)

What is impossible?

- Exceptions
- Input/Output
- State
- Generally: side effects
- $\Rightarrow$ Monads

Kleisli Triples

- An endofunctor $T : \mathcal{C} \to \mathcal{C}$
- A nat. transf. $\eta : \mathrm{id}_{\mathcal{C}} \to T$
- An operator $(-)^* : \mathrm{Hom}_{\mathcal{C}}(X, TY) \to \mathrm{Hom}_{\mathcal{C}}(TX, TY)$

subject to the laws

$$\eta_X^* = \mathrm{id}_{TX} \qquad f^* \circ \eta_X = f \qquad (g^* \circ f)^* = g^* \circ f^*$$

Kleisli Triples

- An endofunctor $T : \mathcal{C} \to \mathcal{C}$
- A nat. transf. $\eta : \mathrm{id}_{\mathcal{C}} \to T$
- An operator $(-)^* : \mathrm{Hom}_{\mathcal{C}}(X, TY) \to \mathrm{Hom}_{\mathcal{C}}(TX, TY)$

subject to the laws

$$\eta_X^* = \mathrm{id}_{TX} \qquad f^* \circ \eta_X = f \qquad (g^* \circ f)^* = g^* \circ f^*$$

Haskell Monads

- A type constructor `M`
- A function `return :: a → M a`
- A function `(>>=) :: M a → (a → M b) → M b`

and laws

```
m >>= return      =  m
return x >>= f    =  f x
(m >>= f) >>= g   =  m >>= (λx → f x >>= g)
```

Alternative Formulation

- An endofunctor $T : \mathcal{C} \to \mathcal{C}$
- A nat. transf. $\mu : T \circ T \to T$
- A nat. transf. $\eta : \mathrm{id}_{\mathcal{C}} \to T$

and laws

- $\mu\,(\mu(T \circ T) \circ T) = \mu\,(T \circ \mu(T \circ T))$
- $\mu\,(\eta(T)) = T = \mu\,(T(\eta))$

With $f^* = \mu_X \circ Tf$ the Kleisli Triple is recovered

Monad examples (in **Sets**)

- State: $TA = S \to (S \times A)$ where S is a set of states
- Exceptions: $TA = E + A$ where E is a set of exceptions
- Non-determinism: $TA = \mathcal{P}A$
- ...

Monad examples (in **Sets**)

- State: $TA = S \to (S \times A)$ where S is a set of states
- Exceptions: $TA = E + A$ where E is a set of exceptions
- Non-determinism: $TA = \mathcal{P}A$
- ...

- $\Rightarrow$ Monad type TA over a type A as an abstraction
- $\Rightarrow$ Add monadic operation to lambda calculus

Concrete examples

in **Sets** ($TA = \mathcal{P}A$)

$$\eta_A x := \{x\} \qquad \mu_A B := \bigcup B$$

Concrete examples

in **Sets** ($TA = \mathcal{P}A$)

$$\eta_A x := \{x\} \qquad \mu_A B := \bigcup B$$

in Haskell ($TA = E + A$ implemented as datatype `Exceptional e a`)

```
data Exceptional e a =
      Success a
    | Exception e

instance Monad (Exceptional e) where
    return              =  Success
    Exception l >>= _   =  Exception l
    Success   r >>= k   =  k r
```

Extension of our Lambda Calculus

Inference Rules

$$\frac{\Gamma \vdash e : A}{\Gamma \vdash \mathsf{ret}\, e : TA}\ \mathsf{ret} \qquad \frac{\Gamma \vdash e : TA \qquad \Gamma, x : A \vdash f : TB}{\Gamma \vdash \mathsf{do}\, x \leftarrow e; f : TB}\ \mathsf{do}$$

Extension of our Lambda Calculus

Inference Rules

$$\frac{\Gamma \vdash e : A}{\Gamma \vdash \mathsf{ret}\, e : TA}\ \mathsf{ret} \qquad \frac{\Gamma \vdash e : TA \qquad \Gamma, x : A \vdash f : TB}{\Gamma \vdash \mathsf{do}\, x \leftarrow e; f : TB}\ \mathsf{do}$$

- ret wraps a value of type A into a computation

Extension of our Lambda Calculus

Inference Rules

$$\frac{\Gamma \vdash e : A}{\Gamma \vdash \mathsf{ret}\, e : TA}\ \mathsf{ret} \qquad \frac{\Gamma \vdash e : TA \qquad \Gamma, x : A \vdash f : TB}{\Gamma \vdash \mathsf{do}\, x \leftarrow e; f : TB}\ \mathsf{do}$$

- ret wraps a value of type A into a computation
- do unwraps a value of type A, but only inside a computation (containment!)

Logical translation

What is the resulting logic?

- Replace the type constructor T by a modal operator $\bigcirc$
- Construct corresponding natural deduction rules

Logical translation

What is the resulting logic?

- Replace the type constructor T by a modal operator $\bigcirc$
- Construct corresponding natural deduction rules

$$\frac{A}{\bigcirc A}\ \bigcirc\mathrm{I}$$

$$\frac{\bigcirc A \qquad \begin{array}{c}\overline{A}\,a\\ \vdots \\ \bigcirc B\end{array}}{\bigcirc B}\ \bigcirc\mathrm{E}_a$$

Lax Modal Logic

Question:

How to interpret $\bigcirc$ logically?

- Possibility?
- Necessity?

Lax Modal Logic

Question:

How to interpret $\bigcirc$ logically?

- Possibility? Problematic (Curry: $\bigcirc A, \bigcirc B \vdash \bigcirc(A \wedge B)$ in Sequent Calculus formulation)
- Necessity?

Lax Modal Logic

Question:

How to interpret $\bigcirc$ logically?

- Possibility? Problematic (Curry: $\bigcirc A, \bigcirc B \vdash \bigcirc(A \wedge B)$ in Sequent Calculus formulation)
- Necessity? Does not agree with intuition

Lax Modal Logic

Question:

How to interpret $\bigcirc$ logically?

- Possibility? Problematic (Curry: $\bigcirc A, \bigcirc B \vdash \bigcirc(A \wedge B)$ in Sequent Calculus formulation)
- Necessity? Does not agree with intuition

Lax Modality:

- Modal operator $\bigcirc$ is a derived notion [Pfenning, Davies]
- $\bigcirc A := \Diamond\Box A$
- $A \to B := \Box A \to B$
- (where the right hand sides are in modal logic with $\Box$ and $\Diamond$)

Lax Modal Logic

Question:

How to interpret $\bigcirc$ logically?

- Possibility? Problematic (Curry: $\bigcirc A, \bigcirc B \vdash \bigcirc(A \wedge B)$ in Sequent Calculus formulation)
- N

Answer:

Monadic λ calculus combines elements of necessity and possibility

Lax M

- Modal operator $\bigcirc$ is a derived notion [Pfenning, Davies]
- $\bigcirc A := \Diamond\Box A$
- $A \to B := \Box A \to B$
- (where the right hand sides are in modal logic with $\Box$ and $\Diamond$)

Dependent Types

- System of *kinds*, *types* have kind $*$
- Lifting of types `a::*` to kinds `{a}`
- Haskell-style data type definition:

```
data Vec :: * → {Nat} → * where
    Nil  ::                    Vec a {Z}
    Cons :: a → Vec a {n} → Vec a {S n}
```

- Index-respecting functions:

```
type s :→ t = ∀ i.s {i} → t {i}
```

Monads

Typeclass

```
class IFunctor m ⇒ IMonad (m::({i} → *) → {i} → *) where
    iskip   ::  p :→ m p
    iextend :: (p :→ m q) → (m p :→ m q)
```

Interpretation

Interpretation

Consider:

- Kinds `p::{i}` $\longrightarrow *$ as *predicates* on `{i}`

Interpretation

Consider:

- Kinds `p::{i}` $\longrightarrow *$ as *predicates* on `{i}`
- Index set `i` as part of a *world state*

Interpretation

Consider:

- Kinds `p::{i}` $\longrightarrow *$ as *predicates* on `{i}`
- Index set `i` as part of a *world state*
- A datum `v::p {i}` as a witness that `p` holds at state `i`

Interpretation

Consider:

- Kinds `p::{i}` $\longrightarrow$ $*$ as *predicates* on `{i}`
- Index set `i` as part of a *world state*
- A datum `v::p {i}` as a witness that `p` holds at state `i`

Then a monad `m` is a predicate transformer expressing *reachability*!

- `m p {i}` asserts reachability of some state satisfying `p` (from state `i`)

Interpretation

Consider:

- Kinds `p::{i}` $\longrightarrow *$ as *predicates* on `{i}`
- Index set `i` as part of a *world state*
- A datum `v::p {i}` as a witness that `p` holds at state `i`

Then a monad `m` is a predicate transformer expressing *reachability*!

- `m p {i}` asserts reachability of some state satisfying `p` (from state `i`)
- `iskip`: if `p` holds, then a state where it holds is reachable by doing nothing

Interpretation

Consider:

- Kinds `p::{i}` $\longrightarrow *$ as *predicates* on `{i}`
- Index set `i` as part of a *world state*
- A datum `v::p {i}` as a witness that `p` holds at state `i`

Then a monad `m` is a predicate transformer expressing *reachability*!

- `m p {i}` asserts reachability of some state satisfying `p` (from state `i`)
- `iskip`: if `p` holds, then a state where it holds is reachable by doing nothing
- `iextend`: `q` reachable from states satisfying `p`
  $\Rightarrow$ `q` reachable from 'here' if `p` is

Conclusion

Kleisli arrows are Hoare Triples

Conclusion

Kleisli arrows are Hoare Triples

- Monad `m` induces a Kleisli category with arrows

$$\texttt{f :: p :⟶ m q}$$

- ⇒ Kleisli arrow says that postcondition `q` is reachable from a precondition `p`
- Composition of Kleisli arrows corresponds to Hoare's sequential composition

```
iseq :: IMonad m ⇒ (p :⟶ m q) → (q :⟶ m r) → p :⟶ m r
iseq f g = iextend g ∘ f
```

- ⇒ Types as specifications in Hoare Logic

References


F. Pfenning and R. Davies.
A Judgmental Reconstruction of Modal Logic (2000)


P.N. Benton, G.M. Bierman and V.C.V. de Paiva
Computational Types from a Logical Perspective (1998)


C. McBride
Kleisli Arrows of Outrageous Fortune (2011)


D. McAdams
A Tutorial on the Curry-Howard Correspondence (2012)


E. Moggi
Notions of Computation and Monads (1989)


M.H.B. Sørensen, P. Urzyczyn
Lectures on the Curry-Howard Isomorphism (1998)