Monads and universal algebra

Anthony Voutas

Australian National University

June, 2012

Introduction

1 Category Theory

Introduction

1. Category Theory
2. Monads

Introduction

1. Category Theory
2. Monads
3. Monads and Adjunctions

Introduction

1. Category Theory
2. Monads
3. Monads and Adjunctions
4. Universal Algebra

Introduction

1. Category Theory
2. Monads
3. Monads and Adjunctions
4. Universal Algebra
5. Comonads

Why Category Theory?

Why Category Theory?

- Algebraic theory of functions

Why Category Theory?

- Algebraic theory of functions
- Alternative language for all mathematics

Why Category Theory?

- Algebraic theory of functions
- Alternative language for all mathematics
- Relatively abstract

Why Category Theory?

- Algebraic theory of functions
- Alternative language for all mathematics
- Relatively abstract
- Highlights relationships missed by other fields

Why Monads?

Why Monads?

- Functional programming

Why Monads?

- Functional programming
- Different view of adjunctions

Why Monads?

- Functional programming
- Different view of adjunctions
- Dual notion: comonad

Why Monads?

- Functional programming
- Different view of adjunctions
- Dual notion: comonad
- Universal algebra

An example with lists

The list monad $\mathbb{T} = \langle T, \mu, \eta \rangle$:
For set X, $T(X)$ set of finite lists of X elements, for functions f and lists $[x_1, x_2, \ldots, x_n] \in TX$:

$$T(f)([x_1, x_2, \ldots, x_n]) = [f(x_1), f(x_2), \ldots, f(x_n)]$$

$\mu_X : T^2(X) \to T(X)$ is a flattening in which:

$$\Big[[x_{1,1}, x_{1,2}, \ldots, x_{1,n}], [x_{2,1}, x_{2,2}, \ldots, x_{2,n}], \ldots, [x_{m,1}, x_{m,2}, \ldots, x_{m,n}]\Big]$$
$$\mapsto \Big[x_{1,1}, x_{1,2}, \ldots, x_{1,n}, x_{2,1}, x_{2,2}, \ldots, x_{2,n}, \ldots, x_{m,1}, x_{m,2}, \ldots, x_{m,n}\Big]$$

and $\eta_X : X \to T(X)$ is an injection $x_1 \mapsto [x_1]$

Monads

A Monad is a triple $\mathbb{T} = \langle T, \mu, \eta \rangle$ such that $T : \mathcal{C} \to \mathcal{C}$ is an endofunctor, μ and η are natural transformations

$$\mu : T^2 \dot{\to} T, \qquad \eta : 1_{\mathcal{C}} \dot{\to} T$$

and the following are commutative diagrams

$$\begin{tikzcd}
T^3 \arrow[r, Rightarrow, "T\mu"] \arrow[d, Rightarrow, "\mu T"'] & T^2 \arrow[d, Rightarrow, "\mu"] \\
T^2 \arrow[r, Rightarrow, "\mu"'] & T
\end{tikzcd}
\qquad
\begin{tikzcd}
1_{\mathcal{C}} T \arrow[r, Rightarrow, "\eta T"] \arrow[dr, equal] & T^2 \arrow[d, Rightarrow, "\mu"] & T 1_{\mathcal{C}} \arrow[l, Rightarrow, "T\eta"'] \arrow[dl, equal] \\
& T &
\end{tikzcd}$$

Monad properties (Informally)

T is an algebra type.

μ is a recipe for taking terms of terms to terms (flattening). Terms of terms of terms flatten to terms unambiguously.

η is a recipe for making singleton terms (embedding). Embedding and then flattening returns the original.

$$\begin{tikzcd} T^3 \arrow[r, Rightarrow, "T\mu"] \arrow[d, Rightarrow, "\mu T"'] & T^2 \arrow[d, Rightarrow, "\mu"] \\ T^2 \arrow[r, Rightarrow, "\mu"'] & T \end{tikzcd} \qquad \begin{tikzcd} 1_{\mathcal{C}} T \arrow[r, Rightarrow, "\eta T"] \arrow[dr, equal] & T^2 \arrow[d, Rightarrow, "\mu"] & T 1_{\mathcal{C}} \arrow[l, Rightarrow, "T\eta"'] \arrow[dl, equal] \\ & T & \end{tikzcd}$$

Free-forgetful adjunction for monoids

Free functor F : **Set** $\to$ **Mon**. $F(X)$ monoid of strings of X elements with concatenation. $Ff([x_1, ..., x_n]) = [f(x_1), ..., f(x_n)]$.

Forgetful functor U : **Mon** $\to$ **Set**. $U(M)$ underlying set of M, monoid homomorphisms sent to underlying functions.

Natural transformations $\eta_X(x_1) = [x_1]$,
$\epsilon_M([m_1, ..., m_n]) = m_1 \circ_M ... \circ_M m_n$.

Monads and Adjunctions

A natural construction for a monad from an adjunction

Theorem

Given an Adjunction (F, G, η, ϵ) with $F \dashv G$, we can always construct a monad in the category $\mathcal{C} = dom(F)$ as follows:

$$T := GF : \mathcal{C} \to \mathcal{C}$$
$$\eta := \eta : 1_{\mathcal{C}} \stackrel{.}{\to} GF$$
$$\mu := G\epsilon F : GFGF \stackrel{.}{\to} GF$$

Monad for monoids

Monad $\langle UF, \eta, U\epsilon F\rangle$. $UF(X)$ the set of lists over X, $\eta_X(x_1) = [x_1]$ and $U\epsilon F_X$ acts as follows:

$$\Big[[x_{1,1}, x_{1,2}, \dots, x_{1,n}], [x_{2,1}, x_{2,2}, \dots, x_{2,n}], \dots, [x_{m,1}, x_{m,2}, \dots, x_{m,n}]\Big]$$

$$\mapsto \Big[x_{1,1}, x_{1,2}, \dots, x_{1,n}, x_{2,1}, x_{2,2}, \dots, x_{2,n}, \dots, x_{m,1}, x_{m,2}, \dots, x_{m,n}\Big]$$

Monad for monoids

Monad $\langle UF, \eta, U\epsilon F\rangle$. $UF(X)$ the set of lists over X, $\eta_X(x_1) = [x_1]$ and $U\epsilon F_X$ acts as follows:

$$\Big[[x_{1,1}, x_{1,2}, \ldots, x_{1,n}], [x_{2,1}, x_{2,2}, \ldots, x_{2,n}], \ldots, [x_{m,1}, x_{m,2}, \ldots, x_{m,n}]\Big]$$
$$\mapsto \Big[x_{1,1}, x_{1,2}, \ldots, x_{1,n}, x_{2,1}, x_{2,2}, \ldots, x_{2,n}, \ldots, x_{m,1}, x_{m,2}, \ldots, x_{m,n}\Big]$$

Same as list monad seen earlier.

Monads to adjunctions?

This raises the question: given a monad, can one split it into two adjoint functors $T = UF$?

Monads to adjunctions?

This raises the question: given a monad, can one split it into two adjoint functors $T = UF$?

Two answers were independently given in 1965. The Kleisli adjunction, and the Eilenberg-Moore adjunction. We will focus on the Eilenberg-Moore adjunction, because it has relevance to universal algebra.

Algebras for a Monad

Objects of The Eilenberg-Moore category $\mathcal{C}^{\mathbb{T}}$ (the $\mathbb{T}$-algebras) will be the pairs (A, a) such that $A \in \mathcal{C}$ and $a : TA \to A$

$$a \circ Ta = a \circ \mu_A \qquad \text{and} \qquad a \circ \eta_A = 1_A$$

That is, the following diagrams commute:

$$\begin{tikzcd} T^2A \arrow[r, "Ta"] \arrow[d, "\mu_A"'] & TA \arrow[d, "a"] \\ TA \arrow[r, "a"'] & A \end{tikzcd} \qquad \begin{tikzcd} A \arrow[r, "\eta_A"] \arrow[dr, equal] & TA \arrow[d, "a"] \\ & A \end{tikzcd}$$

Algebras for the list monad

TA is the set of lists over elements of A, the evaluation a is a function which picks a value from A for any given list.

1. The evaluation a is associative
2. Singleton lists must be evaluated as the value they contain.

These are essentially the requirements of a monoid.

Eilenberg-Moore adjunction

Theorem

Given a monad $\mathbb{T} = \langle T, \mu, \eta \rangle$ *on* $\mathcal{C}$ *we can construct an adjunction between* $\mathcal{C}$ *and* $\mathcal{C}^{\mathbb{T}}$. $U : \mathcal{C}^{\mathbb{T}} \to \mathcal{C}$ *(forgetful)*

$$U(A, a) = A$$
$$U(f : (A, a) \to (B, b)) = f : A \to B$$

and $F : \mathcal{C} \to \mathcal{C}^{\mathbb{T}}$ *(free)*

$$F(A) = (TA, \mu_A)$$
$$F(f : A \to B) = T(f) : (TA, \mu_A) \to (TB, \mu_B)$$

Note: $T = UF$.

EM adjunction for the list monad

$U : \mathbf{Mon} \to \mathbf{Set}$ (forgetful):

$$\begin{aligned} U(M) &= |M| \\ U(h : M \to N) &= h : |M| \to |N| \end{aligned}$$

and $F : \mathbf{Set} \to \mathbf{Mon}$ (free)

$$\begin{aligned} F(A) &= (TA, \mu_A) \\ F(f)([x_1, \ldots, x_n]) &= T(f)([x_1, \ldots, x_n]) \\ &= [f(x_1), \ldots, f(x_n)] \end{aligned}$$

The free-forgetful adjunction seen earlier.

Comparison Functor

Theorem

Given an adjunction $\mathcal{X} \leftrightarrow \mathcal{A}$, there is a functor $K : \mathcal{A} \to \mathcal{X}^{\mathbb{T}}$.

This functor can always be defined. When is K an isomorphism?

Universal Algebra

Universal algebra defines and explores a general concept of algebra. It has produced the notion of an algebraic variety, which is a category of algebras of the same type (groups, rings, monoids, etc).

It is a general result that there are always a free-forgetful adjunction between a variety and **Set**.

Monadicity

An adjunction is called monadic if the comparison functor is an isomorphism.

Beck's Theorem can be used to show that the comparison functor for the free-forgetful adjunction of a variety is an isomorphism.

So a variety is isomorphic to an EM category of algebras.

Duality and comonads

All of these results dualise to the case of comonads. This gives us a coherent notion of coalgebra, which is useful many applications including (but not limited to)

1. modelling state transition systems;
2. reasoning generically about modal logic; and
3. representing unbounded data types (eg streams).

Questions

Questions?