How we use monads without ever realising it

12 March, 2019

Formal languages

How do we picture formal languages and their role in our disciplines?

Formal languages

How do we picture formal languages and their role in our disciplines?

For **philosophy**:

- propositional logic
- first-order logic
- modal logic (+ first-order)
- inductive logic

Formal languages

How do we picture formal languages and their role in our disciplines?

For **philosophy**:

- propositional logic
- first-order logic
- modal logic (+ first-order)
- inductive logic

First-order (untyped) logic as the workhorse for philosophy of language and some metaphysics.

Also modal logic there too – first-order versions trickier.

Formal languages

We also consider:

- Probability theory as a generalisation of logic – Inductive logic – Statistics.
- Mathematics as first-order logic + set theory.
- 'Deviant logics' - intuitionistic, linear, paraconsistent.

Formal languages

We also consider:

- Probability theory as a generalisation of logic – Inductive logic – Statistics.
- Mathematics as first-order logic + set theory.
- 'Deviant logics' - intuitionistic, linear, paraconsistent.

But classical first-order logic is our basic formal language.

Could this picture be overturned?

Could this picture be overturned?

What if there were a different core, one where first-order logic is just a facet.

One allowing much better integration into mathematics, especially that used in physics, and connected with computer science, and even linguistics.

Computational trinitarianism

The central dogma of computational trinitarianism holds that Logic, Languages, and Categories are but three manifestations of one divine notion of computation. There is no preferred route to enlightenment: each aspect provides insights that comprise the experience of computation in our lives.

Computational trinitarianism entails that any concept arising in one aspect should have meaning from the perspective of the other two. If you arrive at an insight that has importance for logic, languages, and categories, then you may feel sure that you have elucidated an essential concept of computation–you have made an enduring scientific discovery.

(Bob Harper)

Computational trinitarianism

propositions as types

+ programs as proofs

+ relationship between type theory and category theory

Computational trinitarianism

propositions as types

+ programs as proofs

+ relationship between type theory and category theory

So that the following are equivalent:

- A proof of a proposition. (In logic.)
- A program with output some type. (In type theory and computer science.)
- A generalized element of an object. (In category theory.)

logic	category theory	type theory
true	terminal object/(-2)-truncated object	h-level 0-type/unit type
false	initial object	empty type
proposition	(-1)-truncated object	h-proposition, mere proposition
proof	generalized element	program
cut rule	composition of classifying morphisms / pullback of display maps	substitution
cut elimination for implication	counit for hom-tensor adjunction	beta reduction
introduction rule for implication	unit for hom-tensor adjunction	eta conversion
logical conjunction	product	product type
disjunction	coproduct ((-1)-truncation of)	sum type (bracket type of)
implication	internal hom	function type
negation	internal hom into initial object	function type into empty type
universal quantification	dependent product	dependent product type
existential quantification	dependent sum ((-1)-truncation of)	dependent sum type (bracket type of)
equivalence	path space object	identity type
equivalence class	quotient	quotient type
induction	colimit	inductive type, W-type, M-type
higher induction	higher colimit	higher inductive type
completely presented set	discrete object/0-truncated object	h-level 2-type/preset/h-set
set	internal 0-groupoid	Bishop set/setoid
universe	object classifier	type of types
modality	closure operator, (idempotent) monad	modal type theory, monad (in computer science)
linear logic	(symmetric, closed) monoidal category	linear type theory/quantum computation
proof net	string diagram	quantum circuit
(absence of) contraction rule	(absence of) diagonal	no-cloning theorem
	synthetic mathematics	domain specific embedded programming language

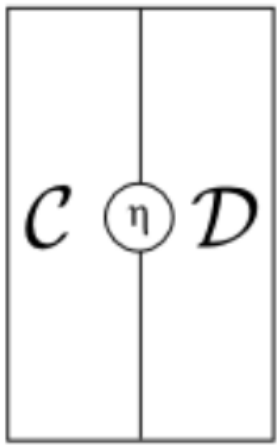
flavor of type theory	equivalent to	flavor of category theory	
intuitionistic propositional logic/simply-typed lambda calculus		cartesian closed category	
multiplicative intuitionistic linear logic		symmetric closed monoidal category	(various authors since ~68)
first-order logic		hyperdoctrine	(Seely 1984a)
classical linear logic		star-autonomous category	(Seely 89)
extensional dependent type theory		locally cartesian closed category	(Seely 1984b)
homotopy type theory without univalence (intensional M-L dependent type theory)		locally cartesian closed (∞,1)-category	(Cisinski 12-(Shulman 12)
homotopy type theory with higher inductive types and univalence		elementary (∞,1)-topos	see here
dependent linear type theory		indexed monoidal category (with comprehension)	(Vákár 14)

Monads in mathematics

Functors are maps between categories, and natural transformations compare functors between the same two categories.

Monads in mathematics

Functors are maps between categories, and natural transformations compare functors between the same two categories.



Monads in mathematics

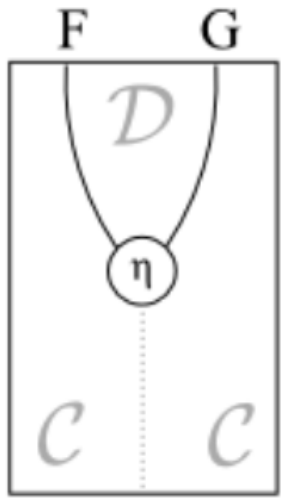
But how about functors in opposite directions?

Some such pairs of functors which are close to being inverses are known as *adjunctions*.

Monads in mathematics

But how about functors in opposite directions?

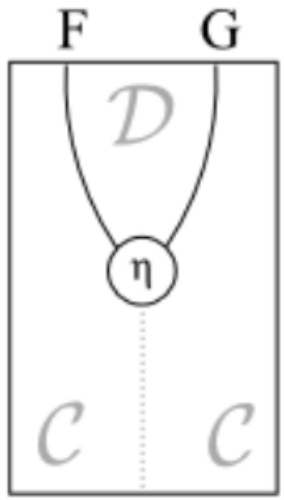
Some such pairs of functors which are close to being inverses are known as *adjunctions*.



Monads in mathematics

But how about functors in opposite directions?

Some such pairs of functors which are close to being inverses are known as *adjunctions*.



A pair of adjoint *functors*, F and G, between two categories $\mathcal{C}$ and $\mathcal{D}$ are such that

$$Hom_{\mathcal{D}}(G(A), B) \cong Hom_{\mathcal{C}}(A, F(B)).$$

Logic and adjoints

We should identify logic with a "*scheme of interlocking adjoints*" (Lawvere 1973, p. 142)

Logic and adjoints

We should identify logic with a "*scheme of interlocking adjoints*" (Lawvere 1973, p. 142)

Take a single category, $\mathcal{C}$, and then form the cartesian product of $\mathcal{C}$ with itself. This is a category which has pairs of objects of $\mathcal{C}$ as objects, and pairs of morphisms of $\mathcal{C}$ as morphisms.

Logic and adjoints

We should identify logic with a "*scheme of interlocking adjoints*" (Lawvere 1973, p. 142)

Take a single category, $\mathcal{C}$, and then form the cartesian product of $\mathcal{C}$ with itself. This is a category which has pairs of objects of $\mathcal{C}$ as objects, and pairs of morphisms of $\mathcal{C}$ as morphisms.

Clearly there is a *diagonal* map, a *functor*, $\mathcal{C} \to \mathcal{C} \times \mathcal{C}$, which sends any object X to (X, X).

We now look to form the right adjoint of this functor.

If it exists, this will have to send a pair of objects of $\mathcal{C}$ to a single object.

Now, given any three objects, A, B, C, in $\mathcal{C}$, we know that

$$Hom_{\mathcal{C}\times\mathcal{C}}((A, A), (B, C)) \cong Hom_{\mathcal{C}}(A, B) \times Hom_{\mathcal{C}}(A, C).$$

So for there to be such an adjunction we need a construction, which we call $B \times C$, such that

$$Hom_{\mathcal{C}}(A, B) \times Hom_{\mathcal{C}}(A, C) \cong Hom_{\mathcal{C}}(A, B \times C).$$

This is precisely what the product construction achieves.

If it exists, this will have to send a pair of objects of $\mathcal{C}$ to a single object.

Now, given any three objects, A, B, C, in $\mathcal{C}$, we know that

$$Hom_{\mathcal{C}\times\mathcal{C}}((A, A), (B, C)) \cong Hom_{\mathcal{C}}(A, B) \times Hom_{\mathcal{C}}(A, C).$$

So for there to be such an adjunction we need a construction, which we call $B \times C$, such that

$$Hom_{\mathcal{C}}(A, B) \times Hom_{\mathcal{C}}(A, C) \cong Hom_{\mathcal{C}}(A, B \times C).$$

This is precisely what the product construction achieves.

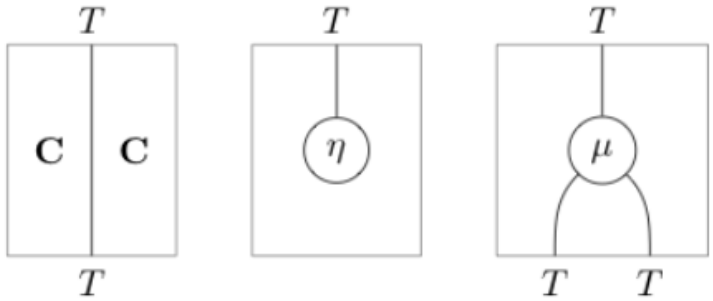
Very similarly, the left adjoint to this diagonal functor corresponds to the sum or *coproduct*

Monads in mathematics

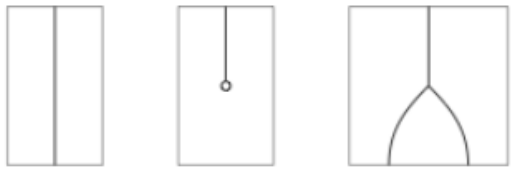
Monads arise by composing such adjunctions. So they act on a single category in a special way.

Monads in mathematics

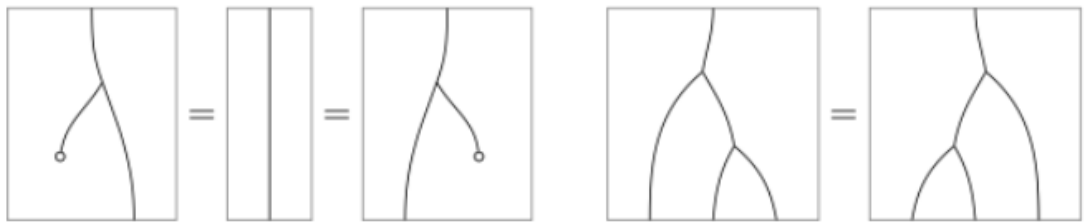
Monads arise by composing such adjunctions. So they act on a single category in a special way.



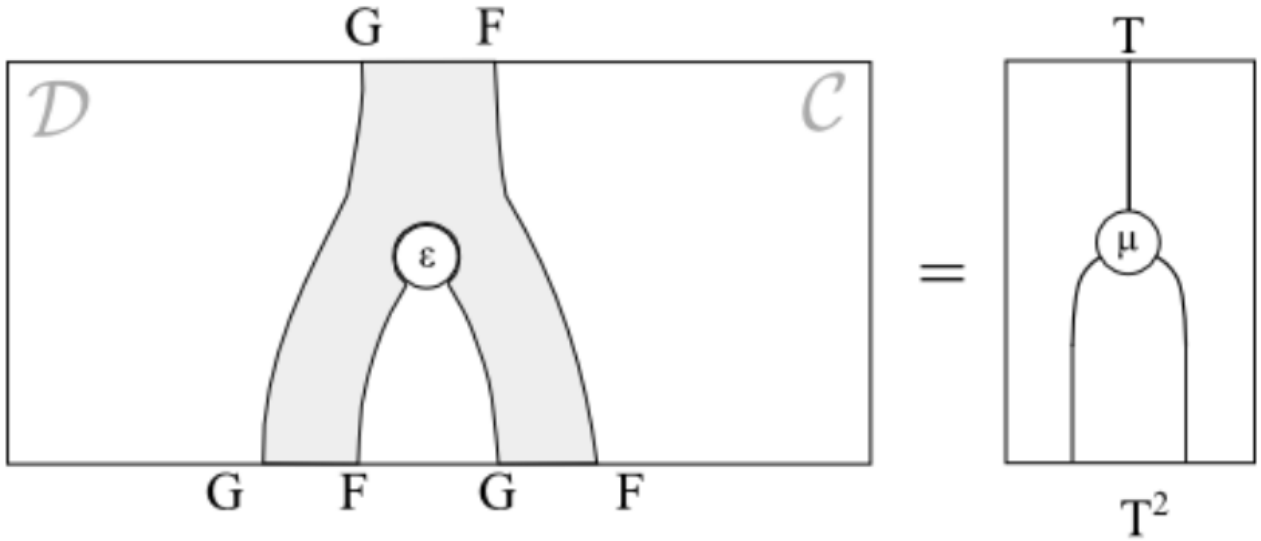
Thanks to the distinctive shapes, one can usually omit the labels:



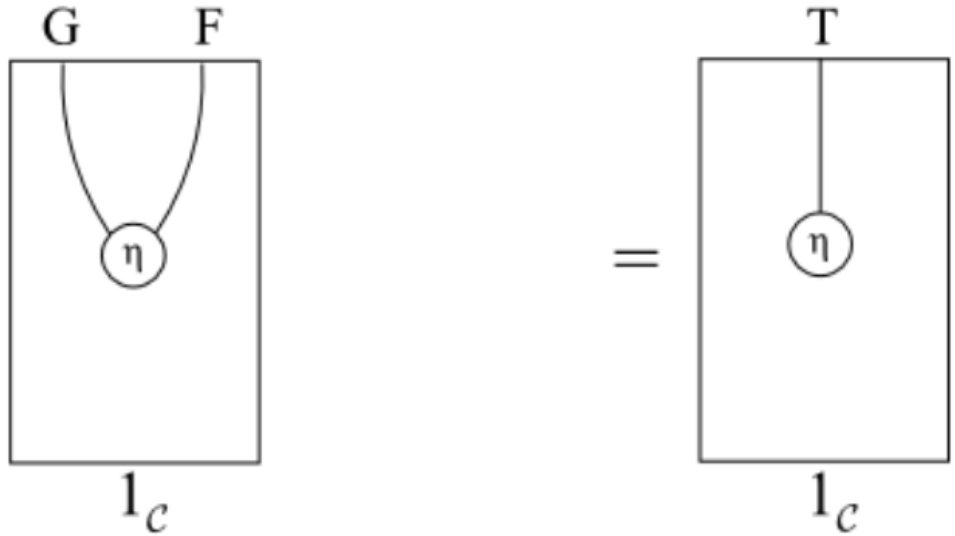
The axioms then appear as:



Monads from adjunctions



$$T = GF$$



Monads and logic

The counit generated by the diagonal-product adjunction is a map in $Hom_{\mathcal{C}\times\mathcal{C}}((A \times B, A \times B), (A, B))$ which represents the paired elimination rules for the product type, and in particular for the conjunction of propositions.

Similarly, the unit of the diagonal-coproduct adjunction represents the paired introduction rules for the sum type.

Synthetic mathematics

(Idempotent) monads are beginning to be used to put geometric flesh on homotopic bones: cohesion, smoothness, supergeometry, equivariance.

This allows modern mathematics to be expressed synthetically (not built up in set theory).

Synthetic mathematics

(Idempotent) monads are beginning to be used to put geometric flesh on homotopic bones: cohesion, smoothness, supergeometry, equivariance.

This allows modern mathematics to be expressed synthetically (not built up in set theory).

And there's also a dependent **linear** type theory for stable homotopy and quantum physics.

Monads in computer science

Meanwhile computer science, in particular the Haskell community of functional programming, has gone wild for monads.

If computing were just the application of function after function what would one know of it?

Consider, for instance, a program that that takes an input, computes something and then writes something else to memory, or sends something to the printer.

Exception monad

Or what about a program that emits an error message during a calculation? How to compose

$$A \to B + E$$

and

$$B \to C + E?$$

Composition

Monads allow you to do this

$$A \to M(B)$$

and

$$B \to M(C)$$

Composition

Monads allow you to do this

$$A \to M(B)$$

and

$$B \to M(C)$$

Then

$$A \to M(B) \to M(M(C)) \to M(C)$$

We see this in the Giry monad, which sends a set to the set of probability distributions on it.

The ‘unit’ map sends elements of a set to the distribution allocating 1 to that element.

Then composition of two conditional distributions is done by integrating a distribution of distributions.

Monads as modal

If A is an object which interprets the values of a particular type, then T(A) is the object which models computation of that type A. [...] On a purely intuitive level and particularly if one thinks about non-termination, there is certainly something appealing about the idea that a computation of type A represents the possibility of a value of type A. (Benton-Bierman-de Paiva 95, p. 1-2)

Here is a short list of similar problems, copied from Eugenio Moggi's seminal paper, all of which are traditionally solved by abandoning the purity of functions.

- Partiality: Computations that may not terminate
- Nondeterminism: Computations that may return many results
- Side effects: Computations that access/modify state
  - Read-only state, or the environment
  - Write-only state, or a log
  - Read/write state
- Exceptions: Partial functions that may fail
- Continuations: Ability to save state of the program and then restore it on demand
- Interactive Input
- Interactive Output

What really is mind blowing is that all these problems may be solved using the same clever trick: turning to embellished functions. Of course, the embellishment will be totally different in each case.

Monads in linguistics

Side effects are to programming languages what pragmatics are to natural languages: they both study how expressions interact with the worlds of their users. It might then come as no surprise that phenomena such as anaphora, presupposition, deixis and conventional implicature yield a monadic description. (Maršík Amblard 2016, p. 3)

Proposed monads

- Conventional implicature: the writer monad
- Optional arguments: the maybe monad
- Presupposition failure: the exception monad
- Anaphora: the state monad and set monad
- Perspective, opacity and intentionality: the reader monad
- Conjunction fallacies: the probability monad
- Quantifier Scope: the continuation monad
- Focus: the pointed set monad
- Interrogatives: the set monad
- Performatives: the IO monad

Reader monad

We have a category, $\mathcal{C}$, and an object, I, so that the slice category, $\mathcal{C}/I$ is sufficiently nice.

Then the lower adjunction generates the reader monad.

$$\mathcal{C}_{/I} \underset{\prod_I}{\overset{\Sigma_I}{\underset{I^*}{\overleftarrow{\longrightarrow}}}} \mathcal{C}$$

Reader monad

We have a category, $\mathcal{C}$, and an object, I, so that the slice category, $\mathcal{C}/I$ is sufficiently nice.

Then the lower adjunction generates the reader monad.

$$\mathcal{C}_{/I} \underset{\prod_I}{\overset{\Sigma_I}{\underset{I^*}{\overleftarrow{\longrightarrow}}}} \mathcal{C}$$

I've spoken before about when I is the type of worlds, and we form monads on the left.

The monad sends an object/type A to A^I.

We can read its elements as terms 'such-and-such according to i'.

Then the monad maps are

- $A \mapsto A^I$, inserting constant value
- $(A^I)^I \mapsto A^I$, x according to i according to $i = x$ according to i

Not only is 'the person downstairs' perspective-relative, but so is 'who i believes is the person downstairs'.

The reader monad, according to Asudeh,

> *"allows us to consider more complex types of meaning only when truly necessary, avoiding the notorious problem of generalizing our lexical entries to the worst case."*

How do we form 'X believes that x is y'?

$X : I, x : A^I, y : A \vdash x(X) = y : Prop$

How do we form 'X believes that x is y'?

$X : I, x : A^I, y : A \vdash x(X) = y : Prop$

May have that $x(X) = a : A$ is true.

x, according to X, is a.

How do we form 'X believes that x is y'?

$X : I, x : A^I, y : A \vdash x(X) = y : Prop$

May have that $x(X) = a : A$ is true.

x, according to X, is a.

We could have y as variable too,

$X : I, x, y : A^I \vdash x(X) = y(X) : Prop$

There could also be a neutral view, a privileged $R : I$, so that $x(R)$ is what x really is.

According to R according to X is according to X.

Since $x(X)$ might be no-one, we could also use the maybe monad, $A + 1$.

de re/de dicto

X believes that the person downstairs is his father.

Substitution is interesting $\vdash x(X) = a, \vdash x(Y) = b$

Y says 'X believes that x is a', so also 'X believes that b is a',

This might result in a sentence like, 'X believes that his mother is his father'.

Asudeh uses this to show how to make sense of

- She loves Peter Parker but not Spiderman.
- He believes π is not π.

Closing thoughts

- Close to Brandom and his vocabulary to specify practice of using base vocabulary.
- There's a field known as adjoint logic of which we can expect great things.

Making explicit aspects of empirical assertion.

- Place, time, other indexicals
- Invariance under change
- Joint assertion
- Selection of assertions. Foregrounding elements of scoresheet.
- Presupposition
- Inference
- Asserter identity, and difference
- Anaphora
- Emotion

Adjoint logic

Being worked out right now (cf. Licata's very recent talk) to capture translations between different inference systems.

Adjoint logic

Being worked out right now (cf. Licata's very recent talk) to capture translations between different inference systems.

> *Curry's proposal was to take $\bigcirc\phi$ as the statement "in some stronger (outer) theory, ϕ holds". As examples of such nested systems of reasoning (with two levels) he suggested Mathematics as the inner and Physics as the outer system, or Physics as the inner system and Biology as the Outer. In both examples the outer system is more encompassing than the inner system where reasoning follows a more rigid notion of truth and deduction. The modality $\bigcirc$, which Curry conceived of as a modality of possibility, is a way of reflecting the relaxed, outer notion of truth within the inner system. (Fairtlough and Mendler 2002, p. 66)*

What of P and 'P' is true? Different entitlements, commitments, incompatibilities?
Is the
What of emphasis? 6 versions of "I didn't take the jam tarts yesterday"
Surely clear as response to a kind of question.

X believes of y that it is P