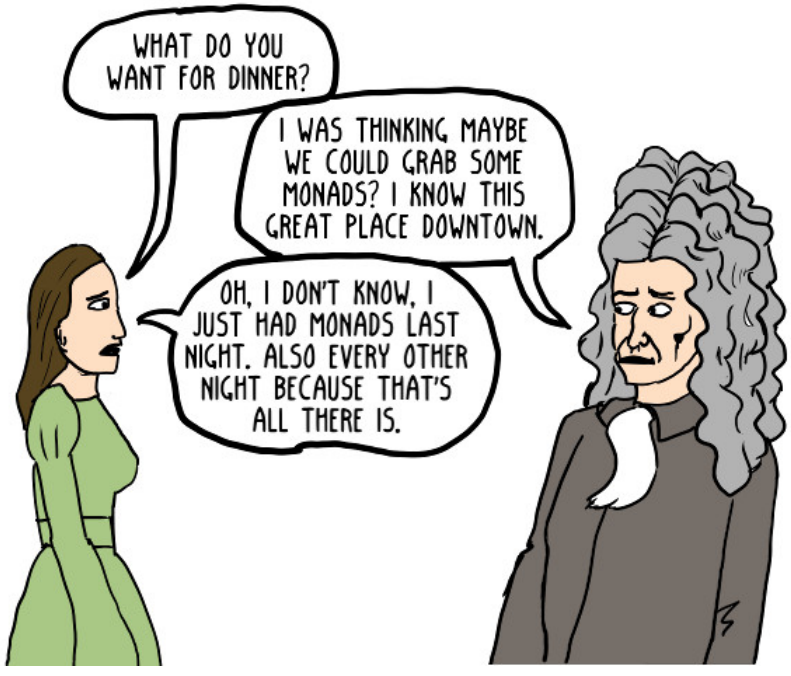
Monads are not what they seem



Tomas Petricek, The Alan Turing Institute
tomasp.net | tomas@tomasp.net | @tomaspetricek

Monad metaphors

How monads are taught

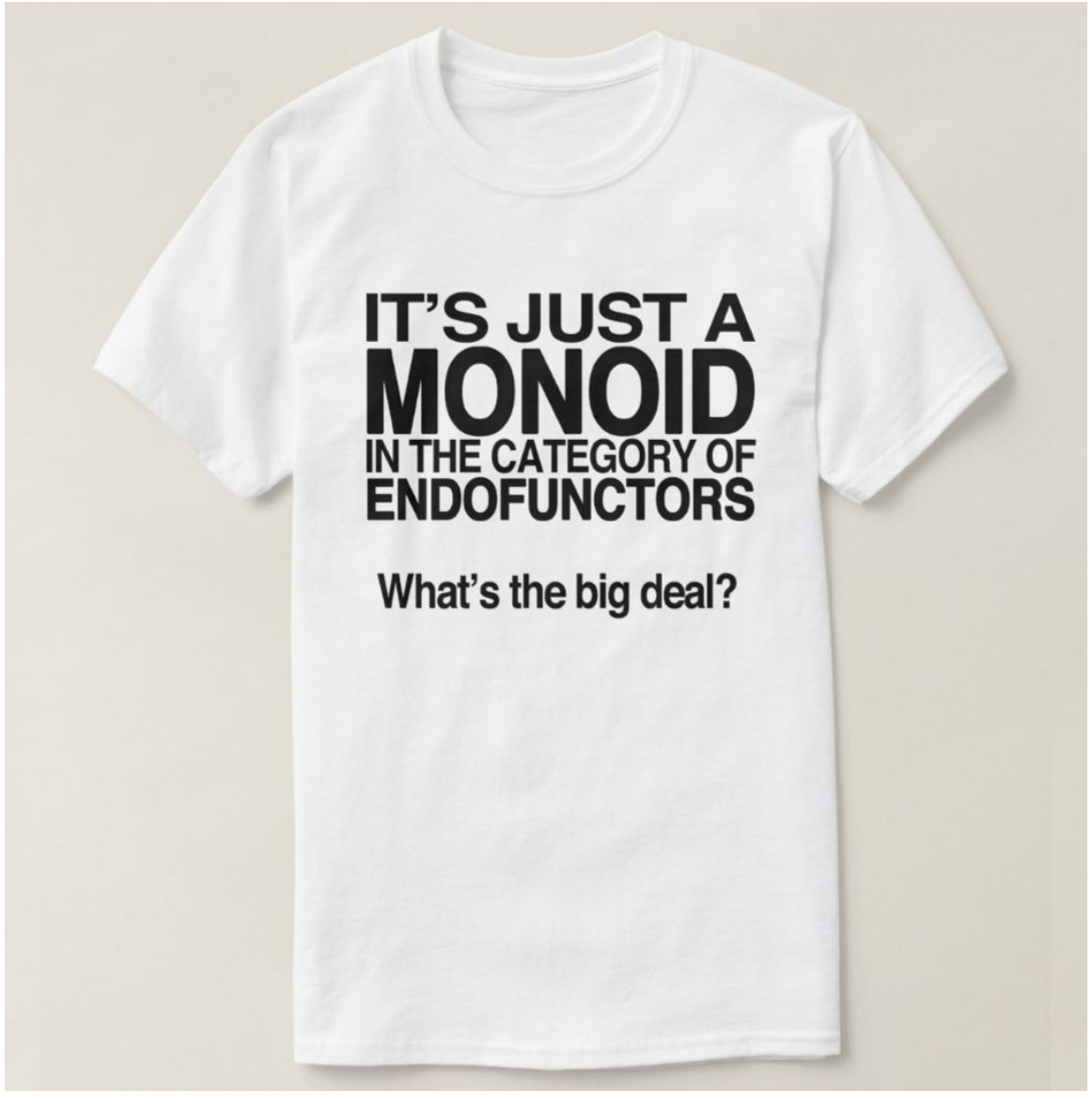
Formal category theory definition

A monad is a functor $M : C \to C$ together with mappings:

- $\eta_\alpha : \alpha \to M\alpha$
- $(-)^*$ turns an arrow $f : \alpha \to M\beta$ into an arrow $f^* : M\alpha \to M\beta$

The mappings are required to satisfy the three monad laws:

- $\eta_\alpha^* = id_{M\alpha}$
- $f^* \circ \eta_\alpha = f$
- $f^* \circ g^* = (f^* \circ g)^*$


IT'S JUST A
MONOID
IN THE CATEGORY OF
ENDOFUNCTORS
What's the big deal?

Monads as generic types

The structure $M\alpha$ represents a data type

- A collection of things - `List int` or `List string`
- A computation - `Lazy int` or `Lazy float`
- You can nest them too - `Lazy (List int)`

Formally: Monads as symbol manipulation

Data type `M a` satisfying monad laws with operations:

- `return` of type `a -> M a`
- `>>=` of type `(a -> M b) -> (M a -> M b)`

Containers: Monads as boxes

`return` wraps thing in a box

`>>=` applies operation on all things in a box

Containers: Monads as burritos

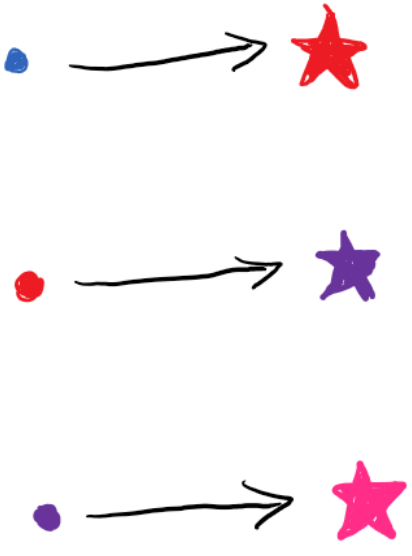
Sequencing: Monads as computations

Plumbing for composing computations with side-effects

Sequencing: Monads as computations

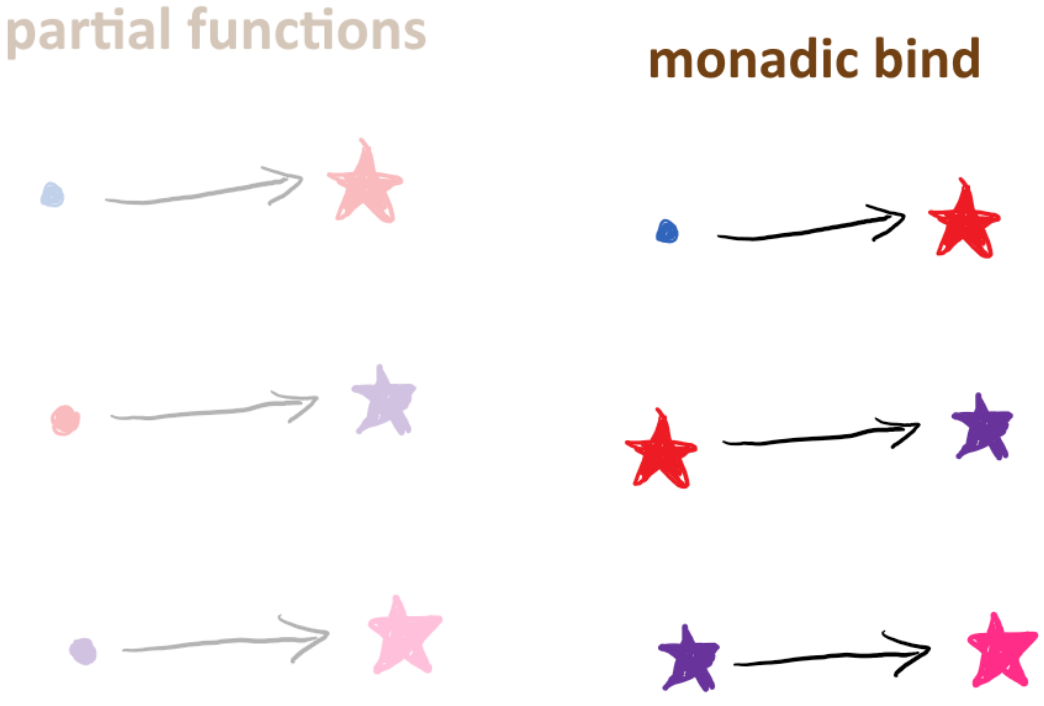
Plumbing for composing computations with side-effects

partial functions



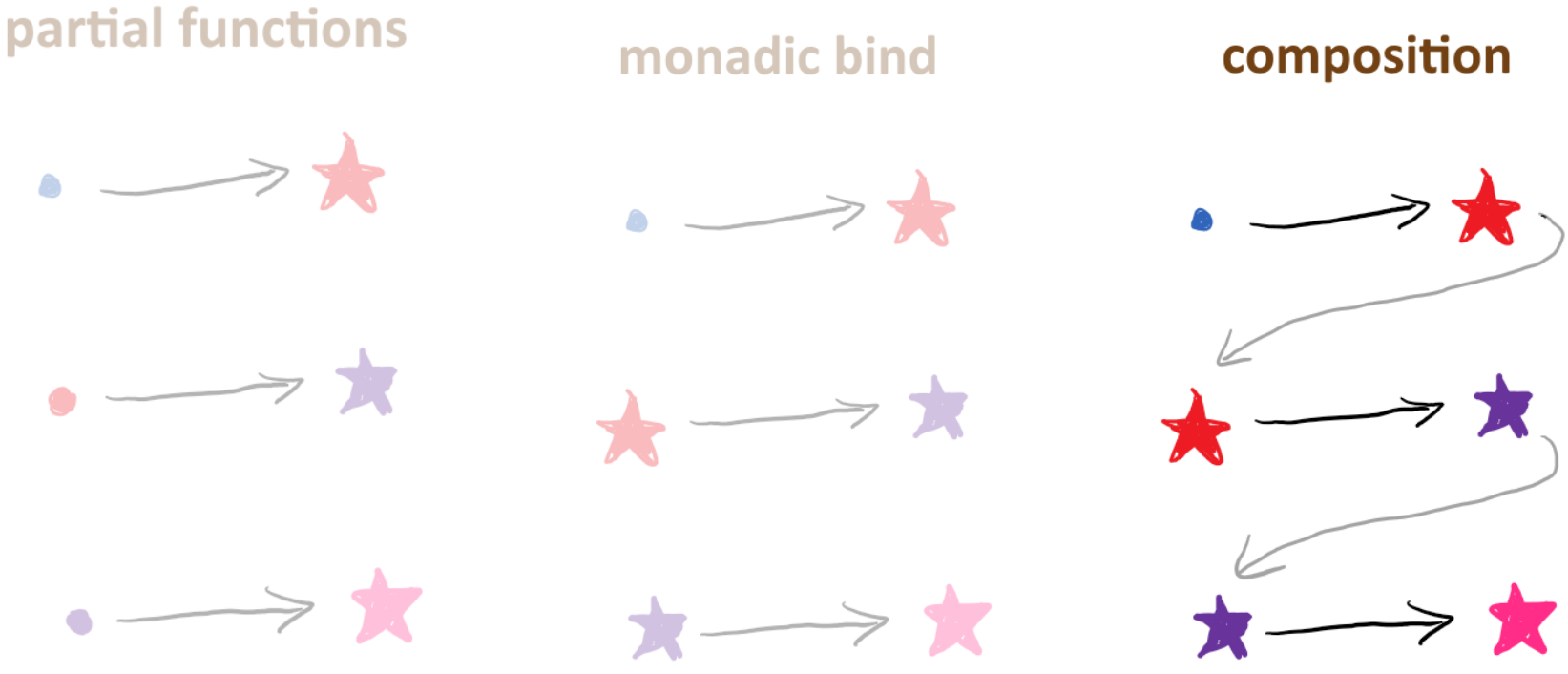
Sequencing: Monads as computations

Plumbing for composing computations with side-effects



Sequencing: Monads as computations

Plumbing for composing computations with side-effects



Metaphors for understanding monads

Neuroscientist's perspective on mathematical thinking

- Movement formal symbol manipulation
- Inside vs. outside for containers and boxes
- Movement for composing comptuations

Monads in practice

How and why people use monads

Syntax for side-effectful comptuations

```
let addr = findAddress "Tomas"
let news = findNews (getCity addr)
return getTop10 news
```

What if `findAddress` and `findNews` can fail?

```
let addr = tryFindAddress "Tomas"
if addr = null then return null
else
  let news = tryFindNews (getCity addr)
  if news = null then return null
  else
    return Success (getTop10 news)
```

Syntax for side-effectful comptuations

```
let addr = findAddress "Tomas"
let news = findNews (getCity addr)
return getTop10 news
```

Monadic notation to remove nesting & repetition:

```
maybe {
  let! addr = tryFindAddress "Tomas"
  let! news = tryFindNews (getCity addr)
  return getTop10 news
}
```

Reasoning about monadic code

Get address and compose message with city:

```
maybe {
  let! addr = findAddress "Tomas"
  return "Tomas lives in " + (getCity addr) }
```

Does extracting `getHome` function change meaning?

```
let getHome name = maybe {
  let! addr = findAddress mame
  return name + " lives in " + getCity addr }
```

```
maybe {
  let! msg = getHome "Tomas"
  return msg }
```

Code reuse using monad abstraction

Write useful functions that work for any monad

- Transform the thing inside the monad

```
mapM : (a -> b) -> (M a -> M b)
```

- Sum list and aggregate side-effects

```
sumM : List (M int) -> M int
```

Misuses of monads

When monads are not the right thing

Monad can be the uninteresting part

The `Par` monad for modelling parallel computations

- Run things in parallel

  ```
  parallel : Par a -> Par b -> Par (a * b)
  ```

- Return the first of two results

  ```
  choose : Par a -> Par a -> Par a
  ```

Also supports monadic `return : a -> Par a`
and `>>= : (a -> Par b) -> (Par a -> Par b)`

Monad as tempting harmful abstraction

`Parser a` reads input string and produces value `a`

- Parse one thing and then another thing
  `Parser a -> Parser b -> Parser (a * b)`
- Try parsing in two ways, use the first success
  `Parser a -> Parser a -> Parser a`

Parsers can be extended to support monadic `>>=` and `return`.

Monad as tempting harmful abstraction

The normal disadvantages of conventional [monadic] parsers, such as their lack of speed and their poor error reporting are remedied.

The techniques [do not] extend to monad-based parsers. [T]he monadic formulation [causes] the evaluation of the parser construction over and over (...).

Deterministic, Error-Correcting Combinator Parsers,
Swierstra & Duponcheel (1996)

Monadic syntax without 'true' monads

Notation originally intended just for monads

Can be used for things that are not monadic

```
H.html {
  H.head {
    H.title "Sample web page"
  }
  H.body {
    H.h1 "Sample web page"
    H.p "This is a sample page..."
  }
}
```

Do we need syntactic flexibility instead of monads?

Philosophy of monads

What monads really are

Concept stretching and changing meaning

Then came the refutationists. In their critical zeal they stretched the concept of polyhedron, to cover objects that were alien to the intended interpretation.

Proofs and refutations,
Lakatos (1979)

Concept stretching and changing meaning

1. Monads are logic for reasoning about effects
2. Language abstraction for encoding effects
3. Abstraction and notation for effects
4. Abstraction and notation (not just) for effects

Monads rooted in research paradigm

One of the goals of the Algol research programme was to utilize the resources of logic to increase the confidence (...) in the correctness of a program.

Science of Operations,
Priestley (2011)

Monads and research paradigms

This paper is about logics for reasoning about programs,
in particular for proving equivalence of programs.

Notions of computations and monads,
Moggi (1991)

Monads and research paradigms

Reasoning about programs often appears in papers

In practice monads are just programming tool

The cost of abstraction is rarely debated

Gentlemen do not talk about syntax

Monads in theory and practice

Before thinking about the philosophy of experiments we should record a certain class or caste difference between the theorizer and the experimenter. It has little to do with philosophy. We find prejudices in favour of theory, as far back as there is institutionalized science.

Representing and intervening,
Hacking (1983)

Anything including monads goes

Galileo replaces one natural interpretation by a very different and as yet at least partly unnatural interpretation. (...) Galileo uses propaganda; he uses psychological tricks, in addition to whatever intellectual reasons he has to offer.

Against method,
Feyerabend (1975)

Anything including monads goes

Community finds monads an attractive topic

They are unnatural until you get them

Popularity means not all uses are justified

Summary

Monads are not what they seem

Monads are not what they seem

Metaphors guide our understanding

Concepts often change their meaning

Paradigms and **propaganda** matter

Applies to other concepts. Bigger picture?

tomasp.net | tomas@tomasp.net | @tomaspetricek