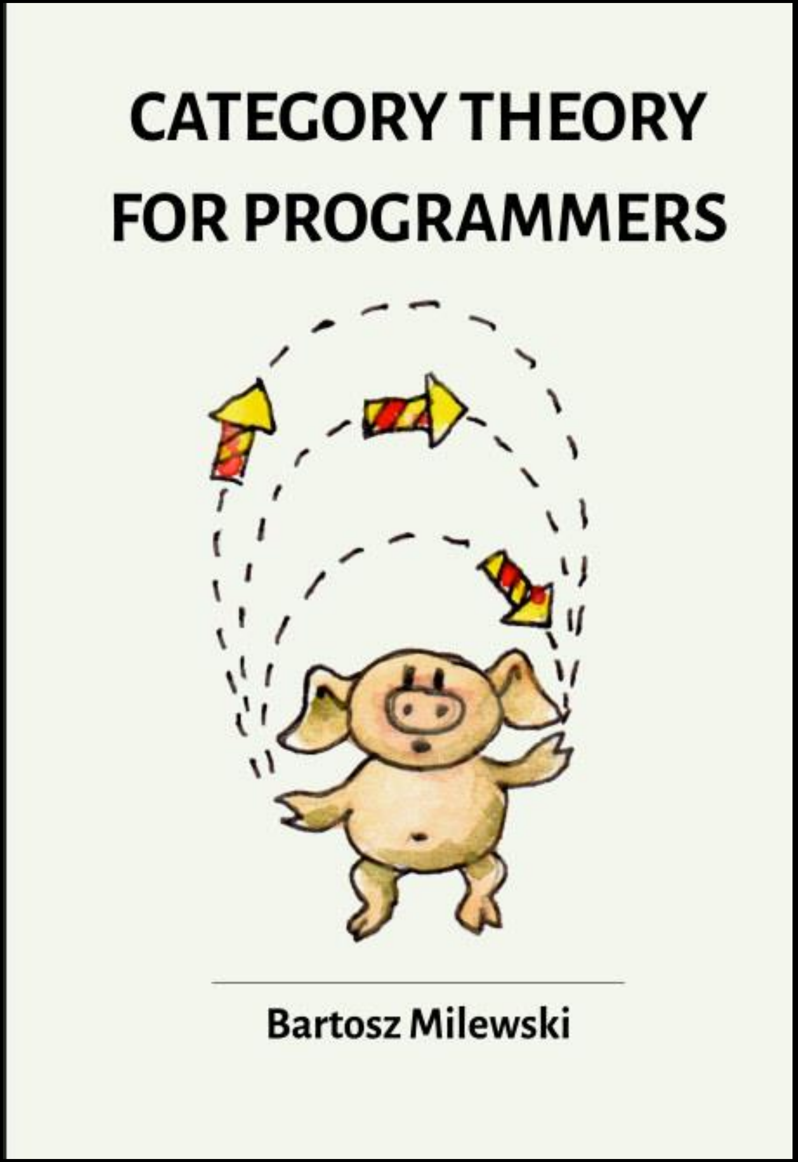
Java

Meetup

Friendly Environment Policy

Berlin Code of Conduct

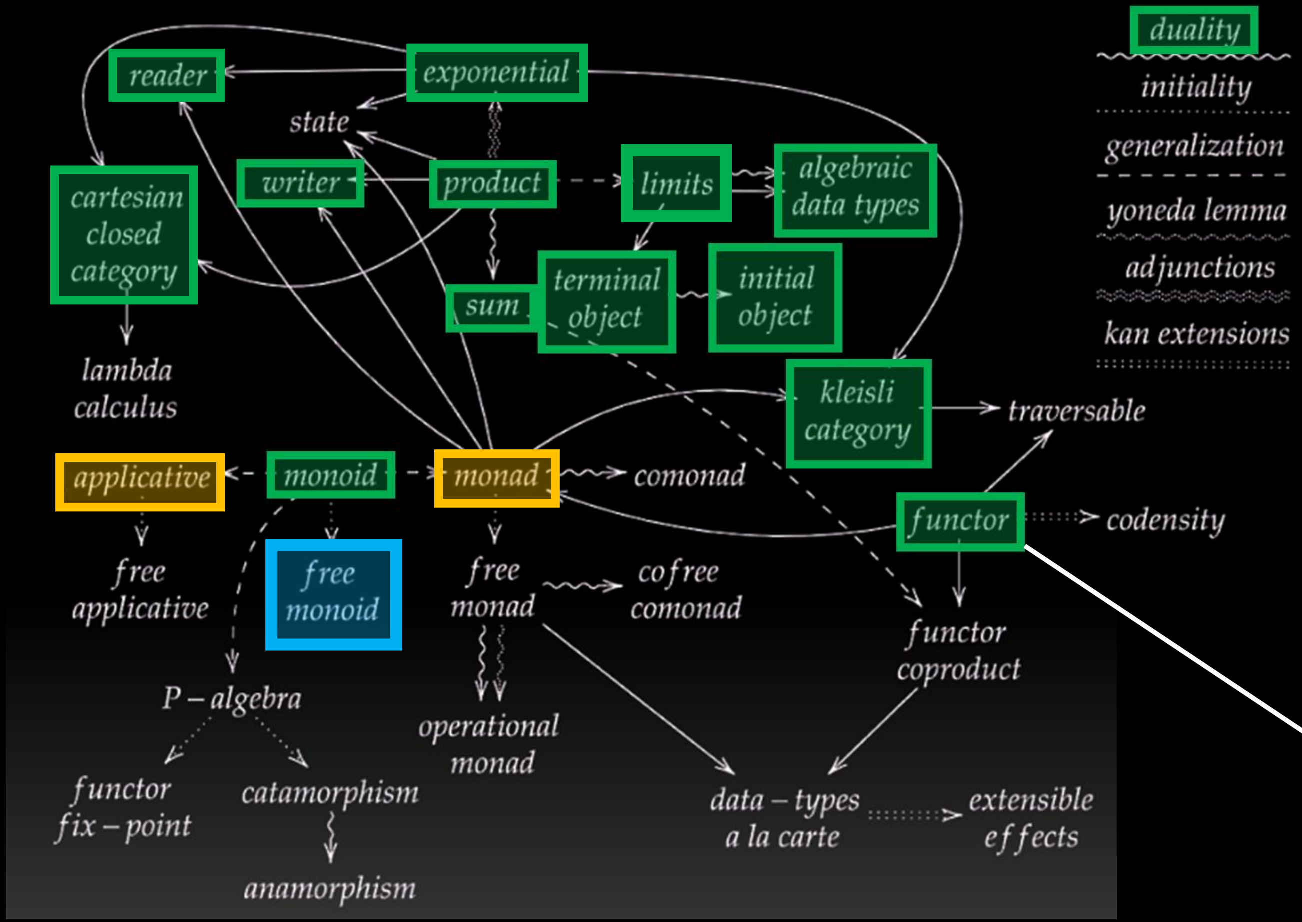


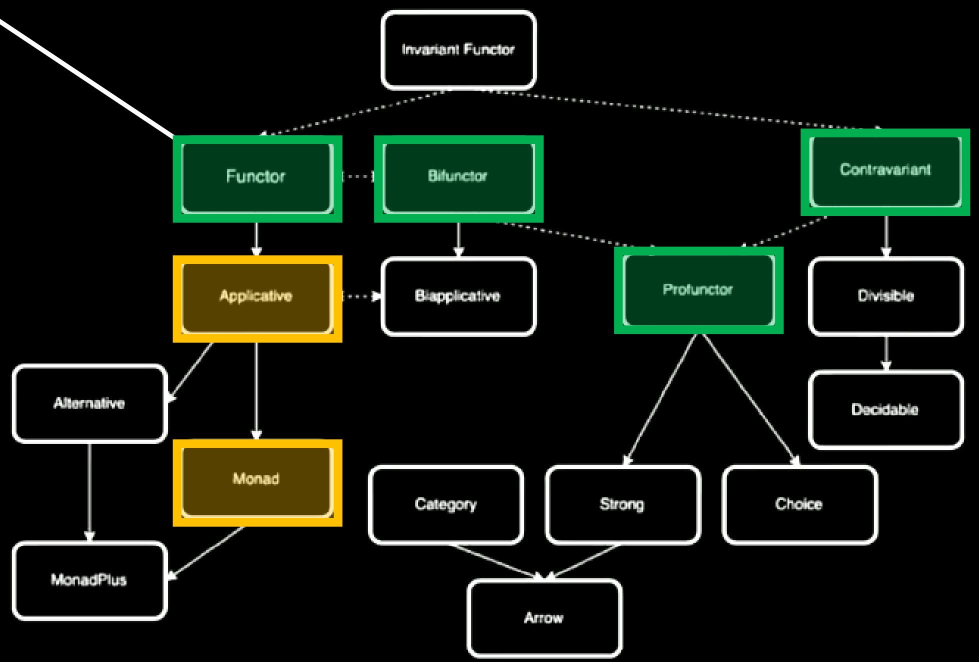
Category Theory for Programmers

Chapter 13:

Free Monoids

The Tools for Thought





MONOIDS ARE AN IMPORTANT concept in both category theory and in programming. Categories correspond to strongly typed languages, monoids to untyped languages. That's because in a monoid you can compose any two arrows, just as in an untyped language you can compose any two functions (of course, you may end up with a runtime error when you execute your program).

Let's see how this works in a simple example. Let's start with a set of two elements, $\{a, b\}$. We'll call them the generators of the free monoid. First, we'll add a special element e to serve as the unit. Next we'll add all the pairs of elements and call them "products". The product of a and b will be the pair (a, b). The product of b and a will be the pair (b, a), the product of a with a will be (a, a), the product of b with b will be (b, b). We can also form pairs with e, like (a, e), (e, b), etc., but we'll identify them with a, b, etc. So in this round we'll only add (a, a), (a, b) and (b, a) and (b, b), and end up with the set $\{e, a, b, (a, a), (a, b), (b, a), (b, b)\}$.

This kind of construction, in which you keep generating all possible combinations of elements, and perform the minimum number of identifications — just enough to uphold the laws — is called a free construction. What we have just done is to construct a *free monoid* from the set of generators $\{a, b\}$.

```haskell
instance Monoid [a] where
    mempty = []
    mappend = (++)
```

It states that an empty list `[]` is the unit element, and list concatenation `(++)` is the binary operation.

Let's first look at monoids as sets equipped with additional structure defined by unit and multiplication. We'll pick as morphisms those functions that preserve the monoidal structure. Such structure-preserving functions are called *homomorphisms*. A monoid homomorphism must map the product of two elements to the product of the mapping of the two elements:

```
h (a * b) = h a * h b
```

and it must map unit to unit.

We'll say that m (together with the function p) is the **free monoid** with the generators x if and only if there is a *unique* morphism h from m to any other monoid n (together with the function q) that satisfies the above factorization property.

$m \qquad \mu :: (m, m) \to m$

$U :: \mathrm{Mon} \to \mathrm{Set}$

$x \quad (m, p :: x \to Um)$

$$\boxed{Uh \circ p = q}$$

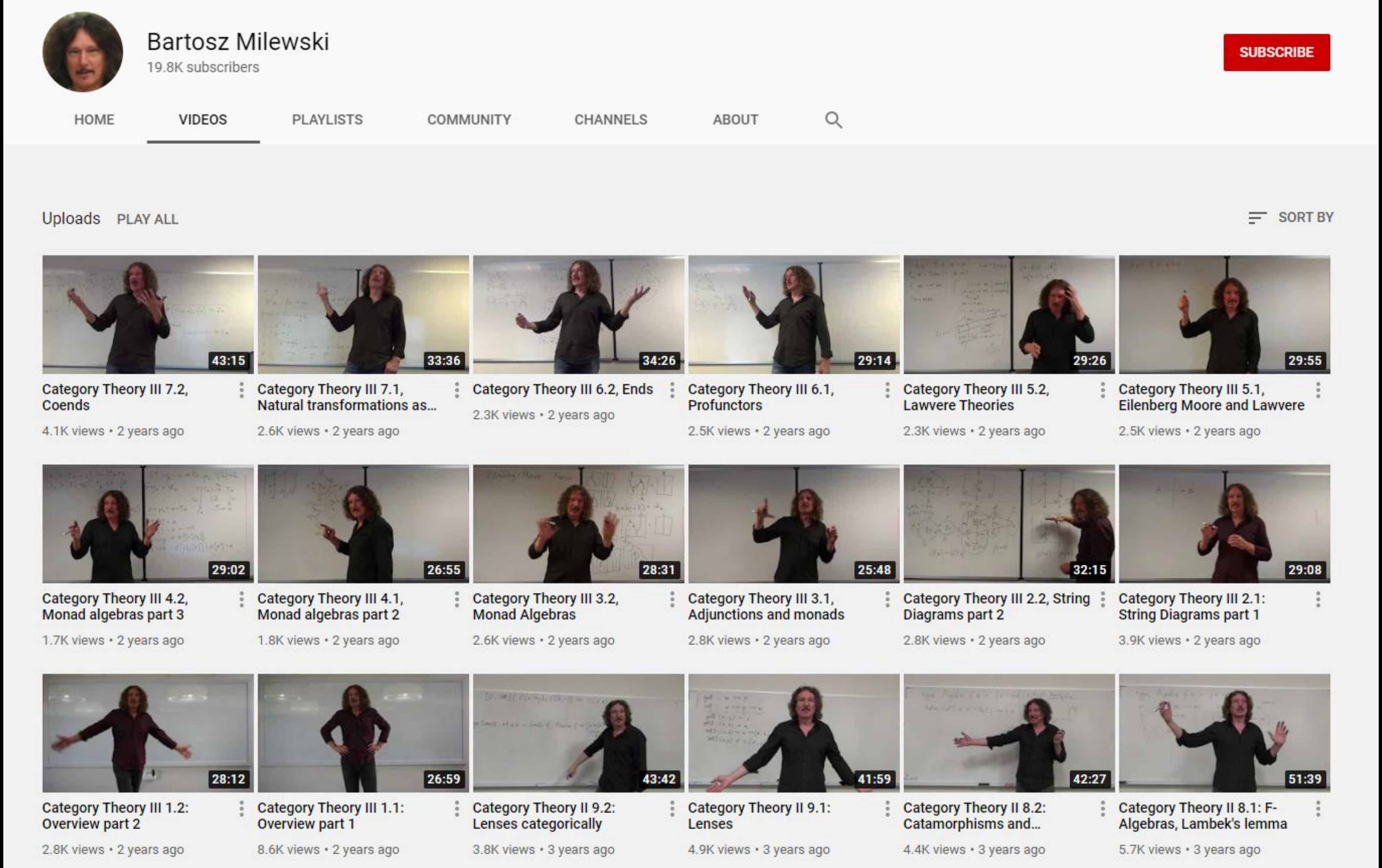
$$\begin{tikzcd}
 & m \arrow[r, "h"] \arrow[d] & n \arrow[d] \\
 & Um \arrow[r, dashed, "Un"] & Un \\
x \arrow[ur, bend left, "p"] \arrow[urr, "q"'] & &
\end{tikzcd}$$

$e * a$ … $e = a$

$(a *$ … $(b * c)$

$h($ … b

2. Consider a monoid homomorphism from lists of integers with concatenation to integers with multiplication. What is the image of the empty list `[]`? Assume that all singleton lists are mapped to the integers they contain, that is `[3]` is mapped to 3, etc. What's the image of `[1, 2, 3, 4]`? How many different lists map to the integer 12? Is there any other homomorphism between the two monoids?



Bartosz Milewski
19.8K subscribers
SUBSCRIBE
HOME
VIDEOS
PLAYLISTS
COMMUNITY
CHANNELS
ABOUT
Uploads
PLAY ALL
SORT BY
43:15
Category Theory III 7.2, Coends
4.1K views • 2 years ago
33:36
Category Theory III 7.1, Natural transformations as...
2.6K views • 2 years ago
34:26
Category Theory III 6.2, Ends
2.3K views • 2 years ago
29:14
Category Theory III 6.1, Profunctors
2.5K views • 2 years ago
29:26
Category Theory III 5.2, Lawvere Theories
2.3K views • 2 years ago
29:55
Category Theory III 5.1, Eilenberg Moore and Lawvere
2.5K views • 2 years ago
29:02
Category Theory III 4.2, Monad algebras part 3
1.7K views • 2 years ago
26:55
Category Theory III 4.1, Monad algebras part 2
1.8K views • 2 years ago
28:31
Category Theory III 3.2, Monad Algebras
2.6K views • 2 years ago
25:48
Category Theory III 3.1, Adjunctions and monads
2.8K views • 2 years ago
32:15
Category Theory III 2.2, String Diagrams part 2
2.8K views • 2 years ago
29:08
Category Theory III 2.1: String Diagrams part 1
3.9K views • 2 years ago
28:12
Category Theory III 1.2: Overview part 2
2.8K views • 2 years ago
26:59
Category Theory III 1.1: Overview part 1
8.6K views • 2 years ago
43:42
Category Theory II 9.2: Lenses categorically
3.8K views • 3 years ago
41:59
Category Theory II 9.1: Lenses
4.9K views • 3 years ago
42:27
Category Theory II 8.2: Catamorphisms and...
4.4K views • 3 years ago
51:39
Category Theory II 8.1: F-Algebras, Lambek's lemma
5.7K views • 3 years ago

Java

Meetup