Java

Meetup

Friendly Environment Policy

Berlin Code of Conduct

Programming Languages Virtual Meetup
1 Tweet

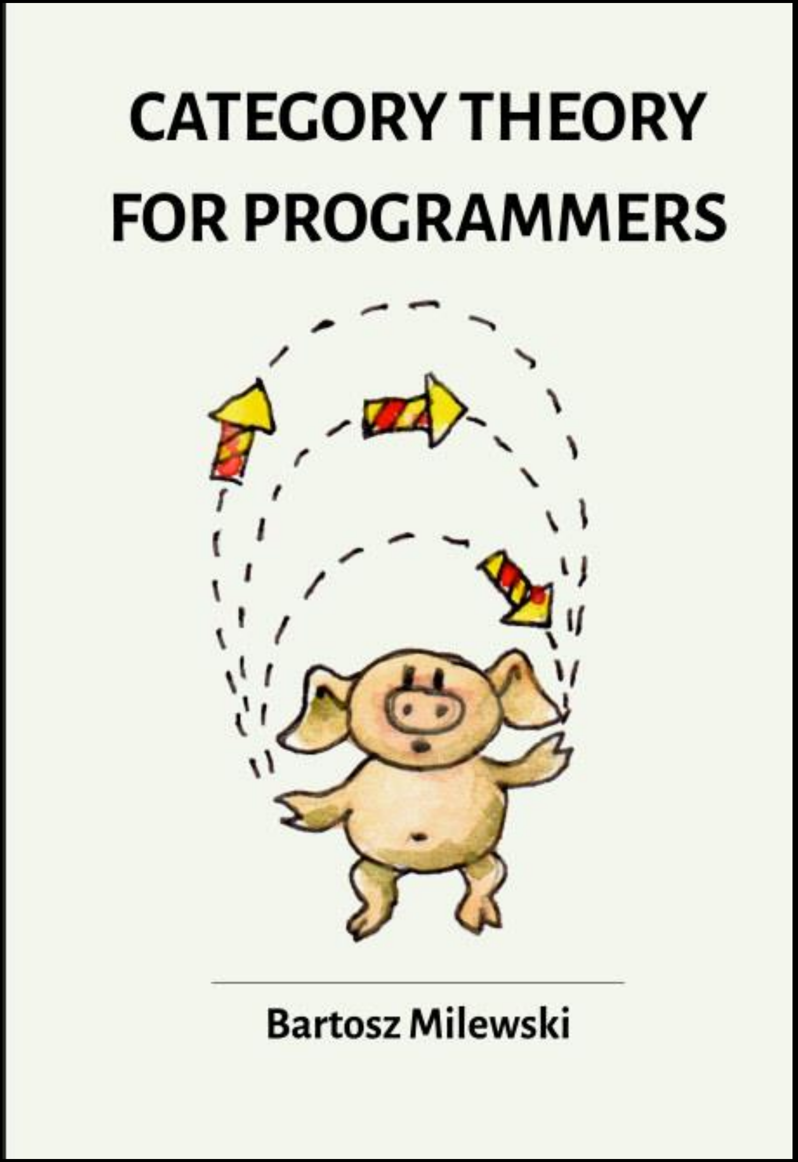
Following

Programming Languages Virtual Meetup
@PLvirtualmeetup

Official Twitter account of the Programming Languages Virtual Meetup. The meetup group is currently working through SICP: web.mit.edu/alexmv/6.037/s....

Toronto, CA meetup.com/Programming-La... Joined March 2020

DISCORD



Category Theory for Programmers

Chapter 14: Representable Functors

It's about time we had a little talk about sets. Mathematicians have a love/hate relationship with set theory. It's the assembly language of mathematics — at least it used to be. Category theory tries to step away from set theory, to some extent. For instance, it's a known fact that the set of all sets doesn't exist, but the category of all sets, **Set**, does. So that's good. On the other hand, we assume that morphisms between any two objects in a category form a set. We even called it a hom-set. To be fair, there is a branch of category theory where morphisms don't form sets. Instead they are objects in another category. Those categories that use hom-objects rather than hom-sets, are called *enriched* categories. In what follows, though, we'll stick to categories with good old-fashioned hom-sets.

Other than that, all the information about sets can be encoded in functions between them — especially the invertible ones called isomorphisms. For all intents and purposes isomorphic sets are identical. Before I summon the wrath of foundational mathematicians, let me explain that the distinction between equality and isomorphism is of fundamental importance. In fact it is one of the main concerns of the latest branch of mathematics, the Homotopy Type Theory (HoTT). I'm mentioning HoTT because it's a pure mathematical theory that takes inspiration from computation, and one of its main proponents, Vladimir Voevodsky, had a major epiphany while studying the Coq theorem prover. The interaction between mathematics and programming goes both ways.

The important lesson about sets is that it's okay to compare sets of unlike elements. For instance, we can say that a given set of natural transformations is isomorphic to some set of morphisms, because a set is just a set. Isomorphism in this case just means that for every natural transformation from one set there is a unique morphism from the other set and vice versa. They can be paired against each other. You can't compare apples with oranges, if they are objects from different categories, but you can compare sets of apples against sets of oranges. Often transforming a categorical problem into a set-theoretical problem gives us the necessary insight or even lets us prove valuable theorems.

Since this construction works in any category, it must also work in the category of Haskell types. In Haskell, the hom-functor is better known as the `Reader` functor:

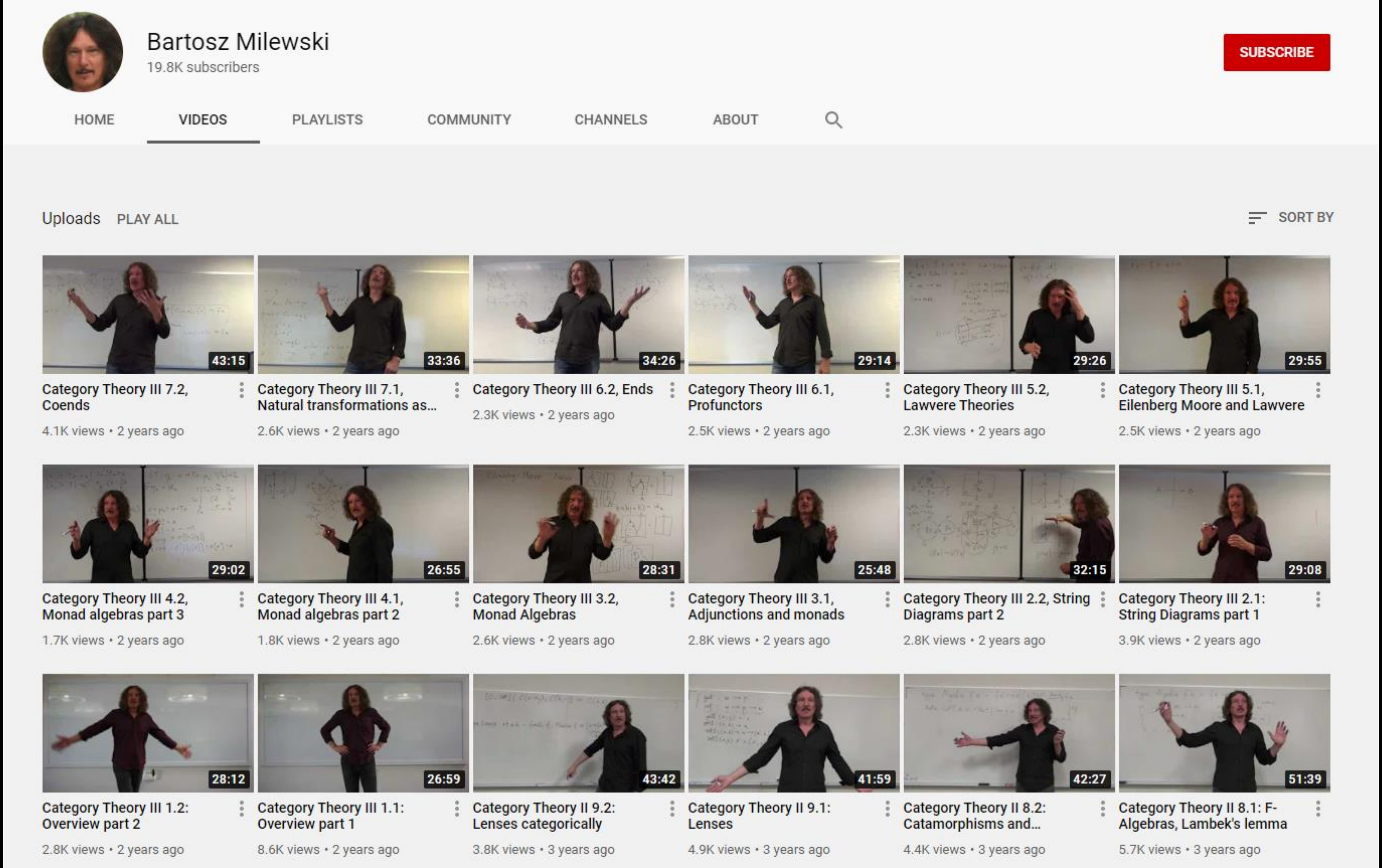
```
type Reader a x = a -> x
```

```
instance Functor (Reader a) where
    fmap f h = f . h
```

```haskell
class Representable f where
    type Rep f :: *
    tabulate :: (Rep f -> x) -> f x
    index    :: f x -> Rep f -> x
```

```haskell
instance Representable Stream where
    type Rep Stream = Integer
    tabulate f = Cons (f 0) (tabulate (f . (+1)))
    index (Cons b bs) n = if n == 0 then b else index bs (n - 1)
```

Of course, this is a purely formal transformation, but if you know some of the properties of logarithms, it is quite helpful. In particular, it turns out that functors that are based on product types can be represented with sum types, and that sum-type functors are not in general representable (example: the list functor).



Bartosz Milewski

19.8K subscribers

SUBSCRIBE

HOME VIDEOS PLAYLISTS COMMUNITY CHANNELS ABOUT

Uploads PLAY ALL SORT BY

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

Java

Meetup