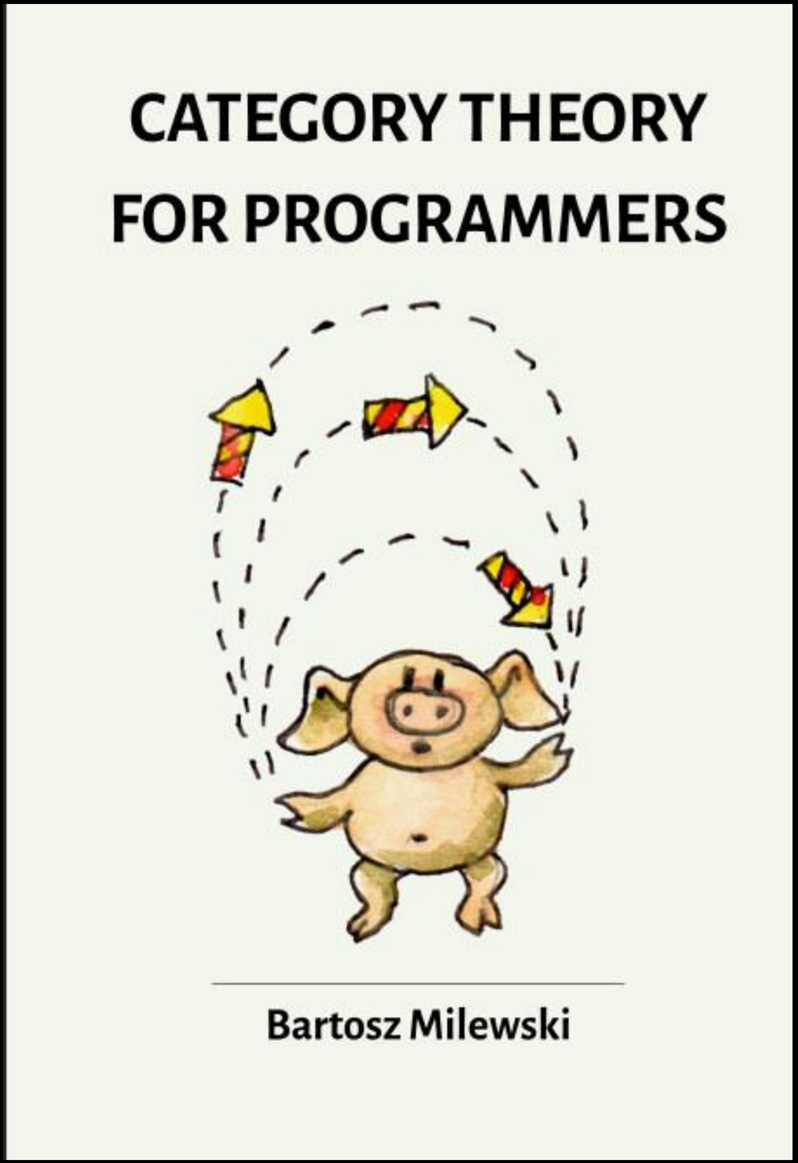
Java

Meetup

Friendly Environment Policy

Berlin Code of Conduct

DISCORD



Category Theory for Programmers

REVIEW #2

Main Concepts	Supporting Concepts
Universal Construction (technique)	
A) **Objects**	Naturality Condition
B) **Morphisms** Also known as “arrows”	Isomorphism
C) **Category** (A + B)	Hom-Set
D) **Functor** (C + B) Bifunctor, Profunctor, Contravariant Functor Product, Coproduct || Maybe, List, Reader	Hom-Functor
E) **Natural Transformation** (D + B)	Yoneda Lemma
	Yoneda Embedding

Main Concepts	Supporting Concepts
Universal Construction (technique)	
A) **Objects**	Naturality Condition
B) **Morphisms** Also known as “arrows”	Isomorphism
C) **Category** (A + B)	Hom-Set
D) **Functor** (C + B) Bifunctor, Profunctor, Contravariant Functor Product, Coproduct || Maybe, List, Reader	Hom-Functor
E) **Natural Transformation** (D + B)	Yoneda Lemma
	Yoneda Embedding

Main Concepts	Supporting Concepts
Universal Construction (technique)	
A) **Objects**	Naturality Condition
B) **Morphisms** Also known as “arrows”	(Natural) Isomorphism
C) **Category** (A + B)	Hom-Set
D) **Functor** (C + B) Bifunctor, Profunctor, Contravariant Functor Product, Coproduct || Maybe, List, Reader	Hom-Functor
E) **Natural Transformation** (D + B)	Yoneda Lemma
	Yoneda Embedding

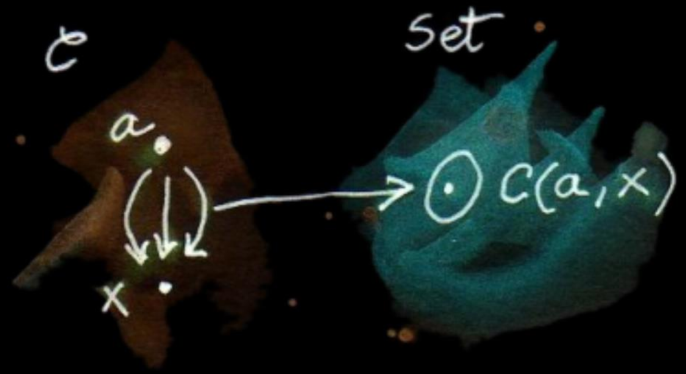
Main Concepts	Supporting Concepts
Universal Construction (technique)	
A) **Objects**	Naturality Condition
B) **Morphisms** Also known as “arrows”	(Natural) Isomorphism
C) **Category** (A + B)	Hom-Set
D) **Functor** (C + B) Bifunctor, Profunctor, Contravariant Functor Product, Coproduct || Maybe, List, Reader	Hom-Functor
E) **Natural Transformation** (D + B)	Representable Functor
	Yoneda Lemma
	Yoneda Embedding

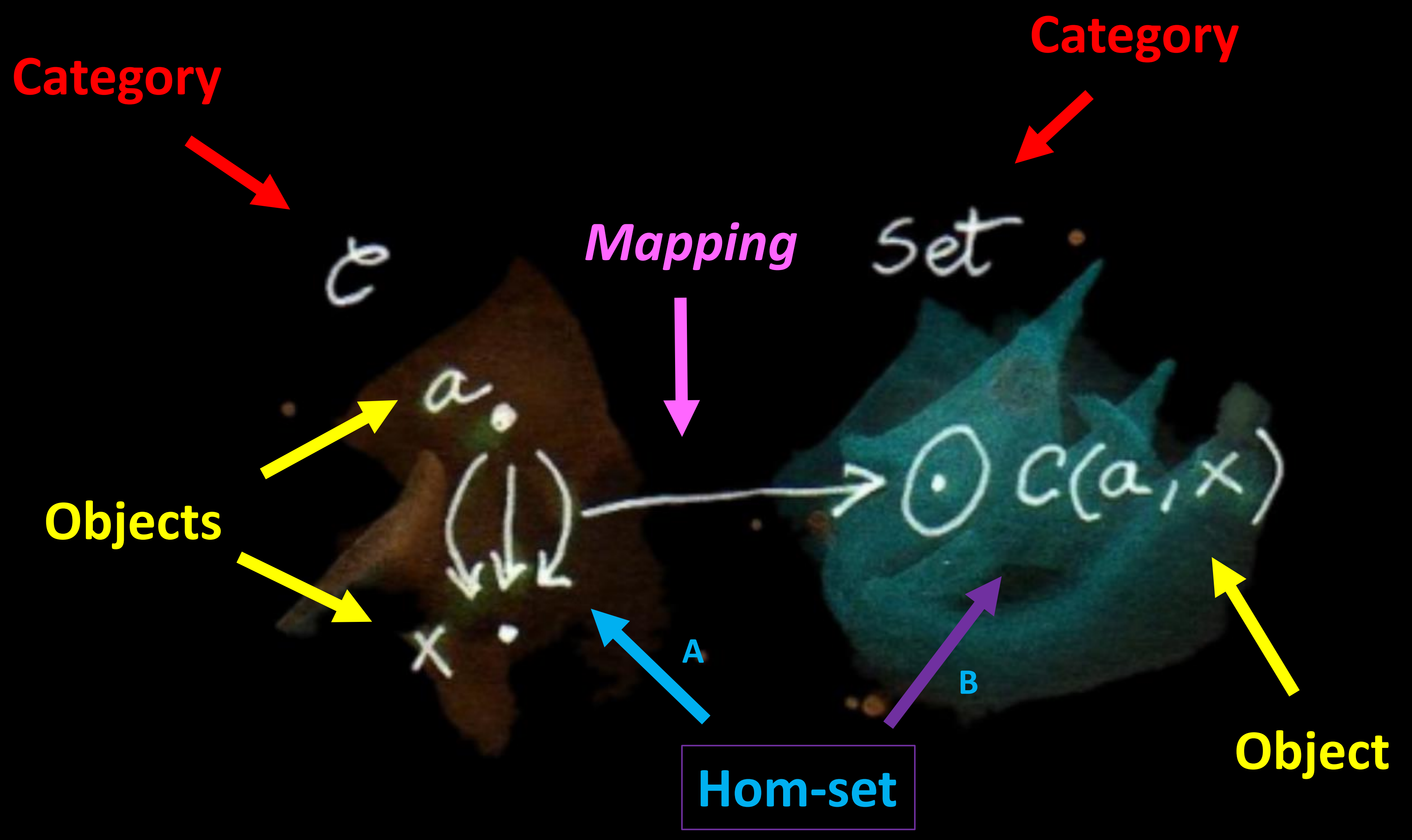
<table>
<tr><th>Main Concepts</th><th>Supporting Concepts</th></tr>
<tr><td colspan="2">Universal Construction (technique)</td></tr>
<tr><td>A) Objects</td><td>Naturality Condition</td></tr>
<tr><td>B) Morphisms
Also known as “arrows”</td><td>(Natural) Isomorphism</td></tr>
<tr><td>C) Category (A + B)</td><td>Hom-Set</td></tr>
<tr><td>D) Functor (C + B)
Bifunctor, Profunctor, Contravariant Functor
Product, Coproduct || Maybe, List, Reader</td><td>Hom-Functor</td></tr>
<tr><td>E) Natural Transformation (D + B)</td><td>Representable Functor</td></tr>
<tr><td></td><td>Yoneda Lemma</td></tr>
<tr><td></td><td>Yoneda Embedding</td></tr>
</table>

14.1 The Hom Functor

Every category comes equipped with a canonical family of mappings to **Set**. Those mappings are in fact functors, so they preserve the structure of the category. Let's build one such mapping.

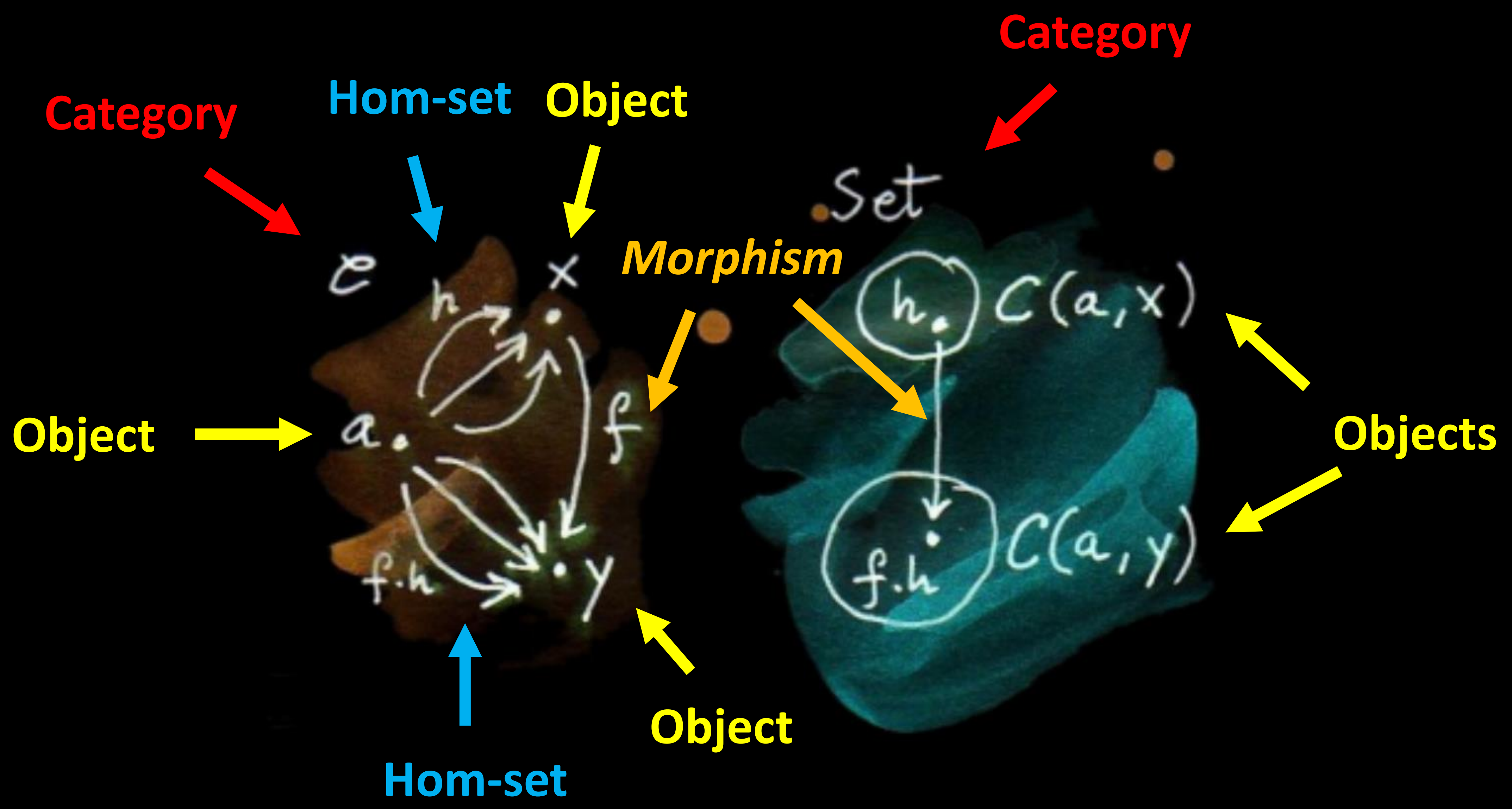
Let's fix one object a in $\mathbf{C}$ and pick another object x also in $\mathbf{C}$. The hom-set $\mathbf{C}(a, x)$ is a set, an object in **Set**. When we vary x, keeping a fixed, $\mathbf{C}(a, x)$ will also vary in **Set**. Thus we have a mapping from x to **Set**.




Category
Category
Mapping
$\mathcal{C}$
Set
a
x
$C(a,x)$
Objects
Object
A
B
Hom-set

If we want to stress the fact that we are considering the hom-set as a mapping in its second argument, we use the notation $\mathbf{C}(a, -)$ with the dash serving as the placeholder for the argument.

This mapping of objects is easily extended to the mapping of morphisms. Let's take a morphism f in $\mathbf{C}$ between two arbitrary objects x and y. The object x is mapped to the set $\mathbf{C}(a, x)$, and the object y is mapped to $\mathbf{C}(a, y)$, under the mapping we have just defined. If this mapping is to be a functor, f must be mapped to a function between the two sets: $\mathbf{C}(a, x) \to \mathbf{C}(a, y)$



Category
Hom-set
Object
Category
Set
Morphism
$\mathcal{C}$
h
x
h
$C(a,x)$
Object
a
f
Objects
$f \cdot h$
y
$f \cdot h$
$C(a,y)$
Object
Hom-set

There two subsections across chapters called **The Hom-Functor**

Ch 8 Functors Section 8

Ch 14 Representable Functors Section 1

The above examples are the reflection of a more general statement that the mapping that takes a pair of objects a and b and assigns to it the set of morphisms between them, the hom-set $\mathbf{C}(a, b)$, is a functor. It is a functor from the product category $\mathbf{C}^{op} \times \mathbf{C}$ to the category of sets, **Set**.

7

Functors

At the risk of sounding like a broken record, I will say this about functors: A functor is a very simple but powerful idea. Category theory is just full of those simple but powerful ideas. A functor is a mapping between categories. Given two categories, **C** and **D**, a functor

Java

Meetup