

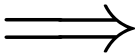
Abstract category
theory

Abstract category
theory

Concrete computations
in computer algebra

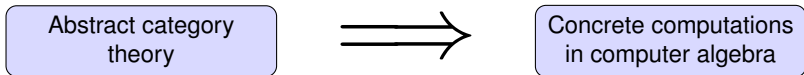
“natural transformation”

Abstract category
theory



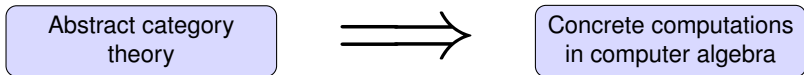
Concrete computations
in computer algebra

“natural transformation”



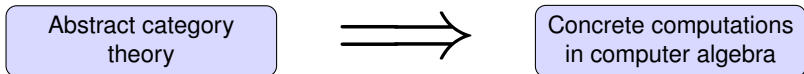
Categorical abstraction

“natural transformation”



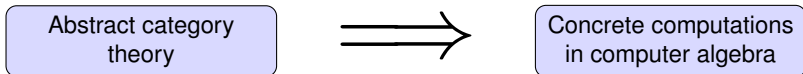
Categorical abstraction is a powerful

“natural transformation”



Categorical abstraction is a powerful organizing principle

“natural transformation”

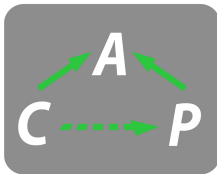


Categorical abstraction is a powerful organizing principle and computational tool.

CAP: Categories, algorithms, programming

Sebastian Posur

August 17, 2019



Categorical abstraction is a powerful **organizing principle** and **computational tool**.

- 1 What is categorical abstraction?

Categorical abstraction is a powerful **organizing principle** and **computational tool**.

- 1 What is categorical abstraction?
- 2 How can it be used as an organizing principle?

Categorical abstraction is a powerful organizing principle and computational tool.

- 1 What is categorical abstraction?
- 2 How can it be used as an organizing principle?
- 3 Why is it a computational tool?

Categorical abstraction is a powerful organizing principle and computational tool.

- 1 What is categorical abstraction?
- 2 How can it be used as an organizing principle?
- 3 Why is it a computational tool?

Categories

Definition

A category \mathcal{A} consists of the following data:

Categories

Definition

A category \mathcal{A} consists of the following data:

- **Objects:** a class $\text{Obj}_{\mathcal{A}}$

A

B

C

Categories

Definition

A category \mathcal{A} consists of the following data:

- **Objects:** a class $\text{Obj}_{\mathcal{A}}$
- **Morphisms:** a family of sets $\text{Hom}_{\mathcal{A}}(A, B)$, where $A, B \in \text{Obj}_{\mathcal{A}}$

A

B

C

Categories

Definition

A category \mathcal{A} consists of the following data:

- **Objects:** a class $\text{Obj}_{\mathcal{A}}$
- **Morphisms:** a family of sets $\text{Hom}_{\mathcal{A}}(A, B)$, where $A, B \in \text{Obj}_{\mathcal{A}}$

$$A \xrightarrow{\alpha} B \qquad C$$

Categories

Definition

A category \mathcal{A} consists of the following data:

- **Objects:** a class $\text{Obj}_{\mathcal{A}}$
- **Morphisms:** a family of sets $\text{Hom}_{\mathcal{A}}(A, B)$, where $A, B \in \text{Obj}_{\mathcal{A}}$

$$A \xrightarrow{\alpha} B \xrightarrow{\beta} C$$

Categories

Definition

A category \mathcal{A} consists of the following data:

- **Objects:** a class $\text{Obj}_{\mathcal{A}}$
- **Morphisms:** a family of sets $\text{Hom}_{\mathcal{A}}(A, B)$, where $A, B \in \text{Obj}_{\mathcal{A}}$
- **Composition:** for $A, B, C \in \text{Obj}_{\mathcal{A}}$, a function
$$\circ : \text{Hom}_{\mathcal{A}}(B, C) \times \text{Hom}_{\mathcal{A}}(A, B) \rightarrow \text{Hom}_{\mathcal{A}}(A, C) \quad (\text{assoc.})$$

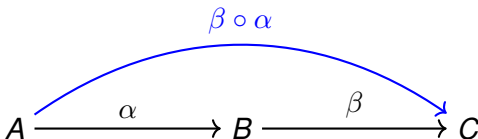
$$A \xrightarrow{\alpha} B \xrightarrow{\beta} C$$

Categories

Definition

A category \mathcal{A} consists of the following data:

- **Objects:** a class $\text{Obj}_{\mathcal{A}}$
- **Morphisms:** a family of sets $\text{Hom}_{\mathcal{A}}(A, B)$, where $A, B \in \text{Obj}_{\mathcal{A}}$
- **Composition:** for $A, B, C \in \text{Obj}_{\mathcal{A}}$, a function
$$\circ : \text{Hom}_{\mathcal{A}}(B, C) \times \text{Hom}_{\mathcal{A}}(A, B) \rightarrow \text{Hom}_{\mathcal{A}}(A, C) \quad (\text{assoc.})$$

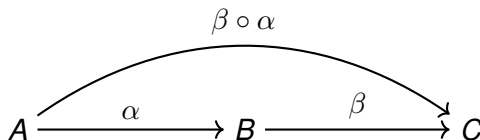


Categories

Definition

A category \mathcal{A} consists of the following data:

- **Objects:** a class $\text{Obj}_{\mathcal{A}}$
- **Morphisms:** a family of sets $\text{Hom}_{\mathcal{A}}(A, B)$, where $A, B \in \text{Obj}_{\mathcal{A}}$
- **Composition:** for $A, B, C \in \text{Obj}_{\mathcal{A}}$, a function
$$\circ : \text{Hom}_{\mathcal{A}}(B, C) \times \text{Hom}_{\mathcal{A}}(A, B) \rightarrow \text{Hom}_{\mathcal{A}}(A, C) \quad (\text{assoc.})$$
- **Neutral elements:** $\text{id}_A \in \text{Hom}_{\mathcal{A}}(A, A)$ (neutral w.r.t. composition)

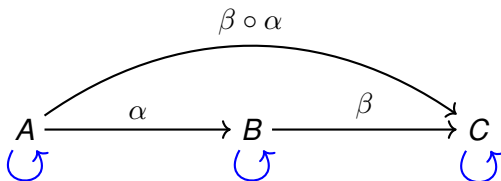


Categories

Definition

A category \mathcal{A} consists of the following data:

- **Objects:** a class $\text{Obj}_{\mathcal{A}}$
- **Morphisms:** a family of sets $\text{Hom}_{\mathcal{A}}(A, B)$, where $A, B \in \text{Obj}_{\mathcal{A}}$
- **Composition:** for $A, B, C \in \text{Obj}_{\mathcal{A}}$, a function
$$\circ : \text{Hom}_{\mathcal{A}}(B, C) \times \text{Hom}_{\mathcal{A}}(A, B) \rightarrow \text{Hom}_{\mathcal{A}}(A, C) \quad (\text{assoc.})$$
- **Neutral elements:** $\text{id}_A \in \text{Hom}_{\mathcal{A}}(A, A)$ (neutral w.r.t. composition)

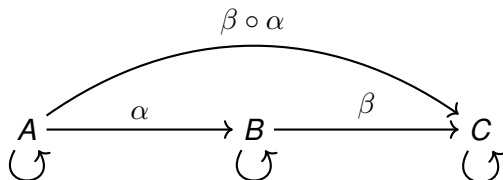


Categories

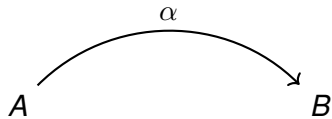
Definition

A category \mathcal{A} consists of the following data:

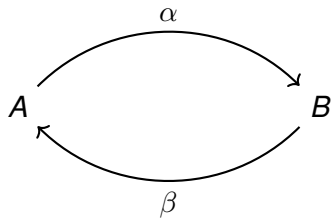
- **Objects:** a class $\text{Obj}_{\mathcal{A}}$
- **Morphisms:** a family of sets $\text{Hom}_{\mathcal{A}}(A, B)$, where $A, B \in \text{Obj}_{\mathcal{A}}$
- **Composition:** for $A, B, C \in \text{Obj}_{\mathcal{A}}$, a function
$$\circ : \text{Hom}_{\mathcal{A}}(B, C) \times \text{Hom}_{\mathcal{A}}(A, B) \rightarrow \text{Hom}_{\mathcal{A}}(A, C) \quad (\text{assoc.})$$
- **Neutral elements:** $\text{id}_A \in \text{Hom}_{\mathcal{A}}(A, A)$ (neutral w.r.t. composition)



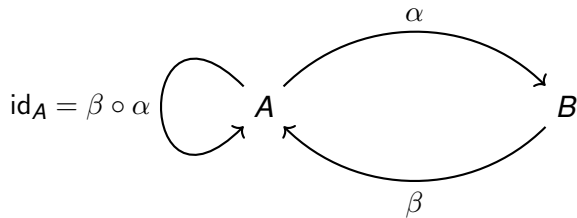
Isomorphisms



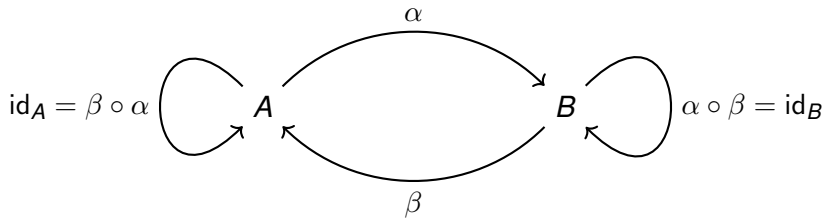
Isomorphisms



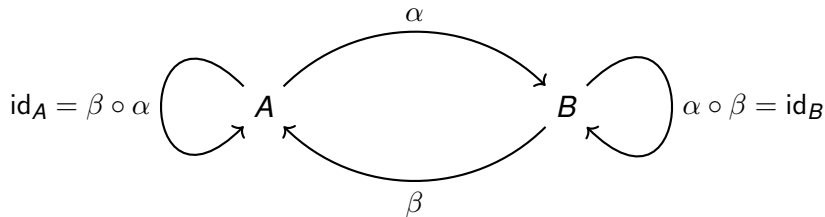
Isomorphisms



Isomorphisms

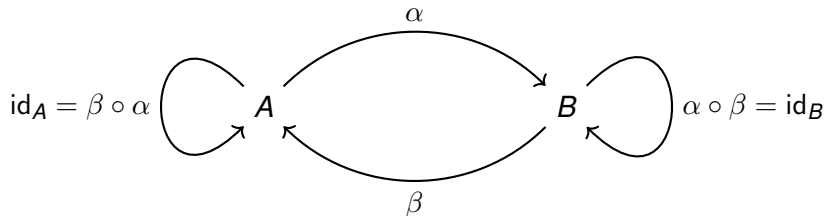


Isomorphisms



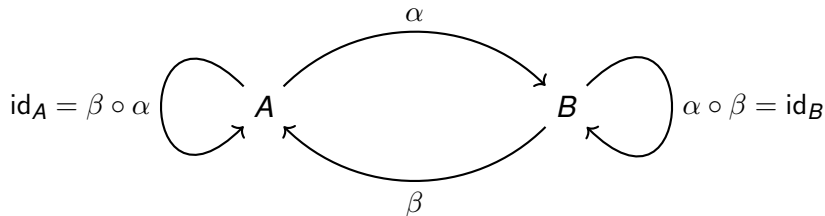
- A and B are isomorphic.

Isomorphisms



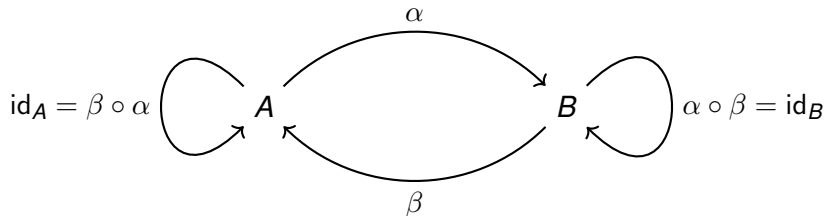
- A and B are isomorphic.
- Notation: $A \cong B$ or $A \simeq B$.

Isomorphisms



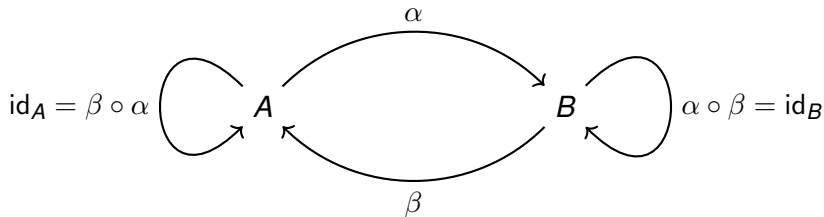
- A and B are isomorphic.
- Notation: $A \cong B$ or $A \simeq B$.
- α is an isomorphism with inverse β .

Isomorphisms



- A and B are isomorphic.
- Notation: $A \cong B$ or $A \simeq B$.
- α is an isomorphism with inverse β .
- β is an isomorphism with inverse α .

Isomorphisms



- A and B are isomorphic.
- Notation: $A \cong B$ or $A \simeq B$.
- α is an isomorphism with inverse β .
- β is an isomorphism with inverse α .

Probably the most important notion in category theory.

Example: Sets

Sets

Sets

- $\text{Obj}_{\text{Sets}} := \{\text{all sets}\}$

Example: Sets

Sets

- $\text{Obj}_{\text{Sets}} := \{\text{all sets}\}$
- $\text{Hom}_{\text{Sets}}(M, N) := \{\text{maps from } M \text{ to } N\}$

Example: Sets

Sets

- $\text{Obj}_{\text{Sets}} := \{\text{all sets}\}$
- $\text{Hom}_{\text{Sets}}(M, N) := \{\text{maps from } M \text{ to } N\}$
- \circ given by composition of maps

Example: Sets

Sets

- $\text{Obj}_{\text{Sets}} := \{\text{all sets}\}$
- $\text{Hom}_{\text{Sets}}(M, N) := \{\text{maps from } M \text{ to } N\}$
- \circ given by composition of maps
- $\text{id}_M : M \longrightarrow M : m \mapsto m$

Example: Sets

Sets

- $\text{Obj}_{\text{Sets}} := \{\text{all sets}\}$
- $\text{Hom}_{\text{Sets}}(M, N) := \{\text{maps from } M \text{ to } N\}$
- \circ given by composition of maps
- $\text{id}_M : M \longrightarrow M : m \mapsto m$

isomorphisms

Example: Sets

Sets

- $\text{Obj}_{\text{Sets}} := \{\text{all sets}\}$
- $\text{Hom}_{\text{Sets}}(M, N) := \{\text{maps from } M \text{ to } N\}$
- \circ given by composition of maps
- $\text{id}_M : M \longrightarrow M : m \mapsto m$

isomorphisms = bijections

Example: Relations

Rels

Example: Relations

Rels

- $\text{Obj}_{\text{Rels}} := \{\text{all sets}\}$

Example: Relations

\mathbf{Rels}

- $\mathbf{Obj}_{\mathbf{Rels}} := \{\text{all sets}\}$
- $\mathbf{Hom}_{\mathbf{Rels}}(M, N) := \{\text{relations from } M \text{ to } N, \text{ i.e., subsets of } M \times N\}$

Example: Relations

\mathbf{Rels}

- $\mathbf{Obj}_{\mathbf{Rels}} := \{\text{all sets}\}$
- $\mathbf{Hom}_{\mathbf{Rels}}(M, N) := \{\text{relations from } M \text{ to } N, \text{ i.e., subsets of } M \times N\}$
- \circ given by composition of relations

Example: Relations

Rels

- $\text{Obj}_{\text{Rels}} := \{\text{all sets}\}$
- $\text{Hom}_{\text{Rels}}(M, N) := \{\text{relations from } M \text{ to } N, \text{ i.e., subsets of } M \times N\}$
- \circ given by composition of relations

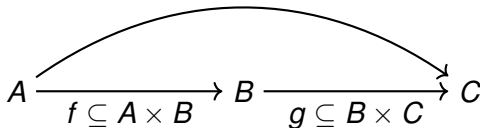
$$A \xrightarrow{f \subseteq A \times B} B \xrightarrow{g \subseteq B \times C} C$$

Example: Relations

\mathbf{Rels}

- $\mathbf{Obj}_{\mathbf{Rels}} := \{\text{all sets}\}$
- $\mathbf{Hom}_{\mathbf{Rels}}(M, N) := \{\text{relations from } M \text{ to } N, \text{ i.e., subsets of } M \times N\}$
- \circ given by composition of relations

$$g \circ f := \{(a, c) \in A \times C \mid \exists b \in B : (a, b) \in f, (b, c) \in g\}$$

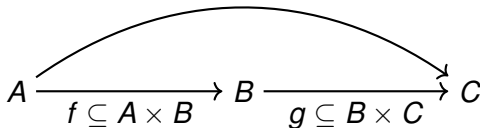


Example: Relations

\mathbf{Rels}

- $\mathbf{Obj}_{\mathbf{Rels}} := \{\text{all sets}\}$
- $\mathbf{Hom}_{\mathbf{Rels}}(M, N) := \{\text{relations from } M \text{ to } N, \text{ i.e., subsets of } M \times N\}$
- \circ given by composition of relations
- $\text{id}_M := \{(m, m) \mid m \in M\} \subset M \times M$

$$g \circ f := \{(a, c) \in A \times C \mid \exists b \in B : (a, b) \in f, (b, c) \in g\}$$



Example: Groups as categories

Let G be a group.

Example: Groups as categories

Let G be a group.

- $\text{Obj}_G := \{*\}$ (any singleton)

Example: Groups as categories

Let G be a group.

- $\text{Obj}_G := \{*\}$ (any singleton)
- $\text{Hom}_G(*, *) := G$

Example: Groups as categories

Let G be a group.

- $\text{Obj}_G := \{*\}$ (any singleton)
- $\text{Hom}_G(*, *) := G$
- \circ given by group multiplication

Example: Groups as categories

Let G be a group.

- $\text{Obj}_G := \{*\}$ (any singleton)
- $\text{Hom}_G(*, *) := G$
- \circ given by group multiplication
- $\text{id}_* :=$ identity element in G

Example: Groups as categories

Let G be a group.

- $\text{Obj}_G := \{*\}$ (any singleton)
- $\text{Hom}_G(*, *) := G$
- \circ given by group multiplication
- $\text{id}_* :=$ identity element in G

isomorphisms

Example: Groups as categories

Let G be a group.

- $\text{Obj}_G := \{*\}$ (any singleton)
- $\text{Hom}_G(*, *) := G$
- \circ given by group multiplication
- $\text{id}_* :=$ identity element in G

isomorphisms = all elements in G

Example: Finite dimensional vector spaces

$\text{vec}_{\mathbb{Q}}$

Example: Finite dimensional vector spaces

$\text{vec}_{\mathbb{Q}}$

- $\text{Obj}_{\text{vec}_{\mathbb{Q}}} := \{\text{all finite dimensional } \mathbb{Q}\text{-vector spaces}\}$

Example: Finite dimensional vector spaces

$\text{vec}_{\mathbb{Q}}$

- $\text{Obj}_{\text{vec}_{\mathbb{Q}}} := \{\text{all finite dimensional } \mathbb{Q}\text{-vector spaces}\}$
- $\text{Hom}_{\text{vec}_{\mathbb{Q}}}(V, W) := \{\mathbb{Q}\text{-linear maps from } V \text{ to } W\}$

Example: Finite dimensional vector spaces

$\text{vec}_{\mathbb{Q}}$

- $\text{Obj}_{\text{vec}_{\mathbb{Q}}} := \{\text{all finite dimensional } \mathbb{Q}\text{-vector spaces}\}$
- $\text{Hom}_{\text{vec}_{\mathbb{Q}}}(V, W) := \{\mathbb{Q}\text{-linear maps from } V \text{ to } W\}$
- \circ given by composition of maps

Example: Finite dimensional vector spaces

$\text{vec}_{\mathbb{Q}}$

- $\text{Obj}_{\text{vec}_{\mathbb{Q}}} := \{\text{all finite dimensional } \mathbb{Q}\text{-vector spaces}\}$
- $\text{Hom}_{\text{vec}_{\mathbb{Q}}}(V, W) := \{\mathbb{Q}\text{-linear maps from } V \text{ to } W\}$
- \circ given by composition of maps
- $\text{id}_M : M \longrightarrow M : m \mapsto m$

Example: Finite dimensional vector spaces

$\text{vec}_{\mathbb{Q}}$

- $\text{Obj}_{\text{vec}_{\mathbb{Q}}} := \{\text{all finite dimensional } \mathbb{Q}\text{-vector spaces}\}$
- $\text{Hom}_{\text{vec}_{\mathbb{Q}}}(V, W) := \{\mathbb{Q}\text{-linear maps from } V \text{ to } W\}$
- \circ given by composition of maps
- $\text{id}_M : M \longrightarrow M : m \mapsto m$

isomorphisms

Example: Finite dimensional vector spaces

$\text{vec}_{\mathbb{Q}}$

- $\text{Obj}_{\text{vec}_{\mathbb{Q}}} := \{\text{all finite dimensional } \mathbb{Q}\text{-vector spaces}\}$
- $\text{Hom}_{\text{vec}_{\mathbb{Q}}}(V, W) := \{\mathbb{Q}\text{-linear maps from } V \text{ to } W\}$
- \circ given by composition of maps
- $\text{id}_M : M \longrightarrow M : m \mapsto m$

isomorphisms = \mathbb{Q} -linear bijections

Example: Matrices

$\text{mat}_{\mathbb{Q}}$

Example: Matrices

$\text{mat}_{\mathbb{Q}}$

- $\text{Obj}_{\text{mat}_{\mathbb{Q}}} := \mathbb{N}_0$

Example: Matrices

$\text{mat}_{\mathbb{Q}}$

- $\text{Obj}_{\text{mat}_{\mathbb{Q}}} := \mathbb{N}_0$
- $\text{Hom}_{\text{vec}_{\mathbb{Q}}}(m, n) := \mathbb{Q}^{m \times n}$

Example: Matrices

$\text{mat}_{\mathbb{Q}}$

- $\text{Obj}_{\text{mat}_{\mathbb{Q}}} := \mathbb{N}_0$
- $\text{Hom}_{\text{vec}_{\mathbb{Q}}}(m, n) := \mathbb{Q}^{m \times n}$
- $A \circ B := B \cdot A$

Example: Matrices

$\text{mat}_{\mathbb{Q}}$

- $\text{Obj}_{\text{mat}_{\mathbb{Q}}} := \mathbb{N}_0$
- $\text{Hom}_{\text{vec}_{\mathbb{Q}}}(m, n) := \mathbb{Q}^{m \times n}$
- $A \circ B := B \cdot A$
- id_m : the $m \times m$ identity matrix

Example: Matrices

$\text{mat}_{\mathbb{Q}}$

- $\text{Obj}_{\text{mat}_{\mathbb{Q}}} := \mathbb{N}_0$
- $\text{Hom}_{\text{vec}_{\mathbb{Q}}}(m, n) := \mathbb{Q}^{m \times n}$
- $A \circ B := B \cdot A$
- id_m : the $m \times m$ identity matrix

isomorphisms

Example: Matrices

$\text{mat}_{\mathbb{Q}}$

- $\text{Obj}_{\text{mat}_{\mathbb{Q}}} := \mathbb{N}_0$
- $\text{Hom}_{\text{vec}_{\mathbb{Q}}}(m, n) := \mathbb{Q}^{m \times n}$
- $A \circ B := B \cdot A$
- id_m : the $m \times m$ identity matrix

isomorphisms = invertible matrices

Equivalences

When are two categories “the same” in a categorical way?

Equivalences

When are two categories “the same” in a categorical way?

 \mathcal{A} \mathcal{B}

Equivalences

When are two categories “the same” in a categorical way?

 \mathcal{A} \mathcal{B}

- $F : \text{Obj}_{\mathcal{A}} \longrightarrow \text{Obj}_{\mathcal{B}}$

Equivalences

When are two categories “the same” in a categorical way?

$$\mathcal{A} \xrightarrow{F} \mathcal{B}$$

- $F : \text{Obj}_{\mathcal{A}} \longrightarrow \text{Obj}_{\mathcal{B}}$

Equivalences

When are two categories “the same” in a categorical way?

$$\mathcal{A} \xrightarrow{F} \mathcal{B}$$

\mathcal{A}

- $F : \text{Obj}_{\mathcal{A}} \longrightarrow \text{Obj}_{\mathcal{B}}$

Equivalences

When are two categories “the same” in a categorical way?

$$\mathcal{A} \xrightarrow{F} \mathcal{B}$$

$$A \longmapsto FA$$

- $F : \text{Obj}_{\mathcal{A}} \longrightarrow \text{Obj}_{\mathcal{B}}$

Equivalences

When are two categories “the same” in a categorical way?

$$\mathcal{A} \xrightarrow{F} \mathcal{B}$$

$$A \longmapsto FA$$

- $F : \text{Obj}_{\mathcal{A}} \longrightarrow \text{Obj}_{\mathcal{B}}$
- $\text{Hom}_{\mathcal{A}}(A, A') \longrightarrow \text{Hom}_{\mathcal{B}}(FA, FA')$

Equivalences

When are two categories “the same” in a categorical way?

$$\mathcal{A} \xrightarrow{F} \mathcal{B}$$

$$\begin{array}{c} A \\ \alpha \downarrow \\ A' \end{array}$$

- $F : \text{Obj}_{\mathcal{A}} \longrightarrow \text{Obj}_{\mathcal{B}}$
- $\text{Hom}_{\mathcal{A}}(A, A') \longrightarrow \text{Hom}_{\mathcal{B}}(FA, FA')$

Equivalences

When are two categories “the same” in a categorical way?

$$\mathcal{A} \xrightarrow{F} \mathcal{B}$$

$$\begin{array}{ccc} A & & FA \\ \alpha \downarrow & \mapsto & \downarrow F\alpha \\ A' & & FA' \end{array}$$

- $F : \text{Obj}_{\mathcal{A}} \longrightarrow \text{Obj}_{\mathcal{B}}$
- $\text{Hom}_{\mathcal{A}}(A, A') \longrightarrow \text{Hom}_{\mathcal{B}}(FA, FA')$

Equivalences

When are two categories “the same” in a categorical way?

$$\mathcal{A} \xrightarrow{F} \mathcal{B}$$

$$\begin{array}{ccc} A & & FA \\ \alpha \downarrow & \mapsto & \downarrow F\alpha \\ A' & & FA' \end{array}$$

- $F : \text{Obj}_{\mathcal{A}} \longrightarrow \text{Obj}_{\mathcal{B}}$
- $\text{Hom}_{\mathcal{A}}(A, A') \longrightarrow \text{Hom}_{\mathcal{B}}(FA, FA') \quad (\text{respects id and } \circ, \text{ bijection})$

Equivalences

When are two categories “the same” in a categorical way?

$$\mathcal{A} \xrightarrow{F} \mathcal{B}$$

$$\begin{array}{ccc} A & & FA \\ \alpha \downarrow & \mapsto & \downarrow F\alpha \\ A' & & FA' \end{array}$$

- $F : \text{Obj}_{\mathcal{A}} \longrightarrow \text{Obj}_{\mathcal{B}}$
- $\text{Hom}_{\mathcal{A}}(A, A') \longrightarrow \text{Hom}_{\mathcal{B}}(FA, FA')$ (respects id and \circ , bijection)
- $\forall B \in \mathcal{B}$

Equivalences

When are two categories “the same” in a categorical way?

$$\mathcal{A} \xrightarrow{F} \mathcal{B}$$

- $F : \text{Obj}_{\mathcal{A}} \longrightarrow \text{Obj}_{\mathcal{B}}$
- $\text{Hom}_{\mathcal{A}}(A, A') \longrightarrow \text{Hom}_{\mathcal{B}}(FA, FA')$ (respects id and \circ , bijection)
- $\forall B \in \mathcal{B}$

Equivalences

When are two categories “the same” in a categorical way?

$$\mathcal{A} \xrightarrow{F} \mathcal{B}$$

\mathcal{B}

- $F : \text{Obj}_{\mathcal{A}} \longrightarrow \text{Obj}_{\mathcal{B}}$
- $\text{Hom}_{\mathcal{A}}(A, A') \longrightarrow \text{Hom}_{\mathcal{B}}(FA, FA')$ (respects id and \circ , bijection)
- $\forall B \in \mathcal{B}$

Equivalences

When are two categories “the same” in a categorical way?

$$\mathcal{A} \xrightarrow{F} \mathcal{B}$$

\mathcal{B}

- $F : \text{Obj}_{\mathcal{A}} \longrightarrow \text{Obj}_{\mathcal{B}}$
- $\text{Hom}_{\mathcal{A}}(A, A') \longrightarrow \text{Hom}_{\mathcal{B}}(FA, FA')$ (respects id and \circ , bijection)
- $\forall B \in \mathcal{B} \quad \exists A \in \mathcal{A} :$

Equivalences

When are two categories “the same” in a categorical way?

$$\mathcal{A} \xrightarrow{F} \mathcal{B}$$

\mathcal{A}

\mathcal{B}

- $F : \text{Obj}_{\mathcal{A}} \longrightarrow \text{Obj}_{\mathcal{B}}$
- $\text{Hom}_{\mathcal{A}}(A, A') \longrightarrow \text{Hom}_{\mathcal{B}}(FA, FA')$ (respects id and \circ , bijection)
- $\forall B \in \mathcal{B} \quad \exists A \in \mathcal{A} :$

Equivalences

When are two categories “the same” in a categorical way?

$$\mathcal{A} \xrightarrow{F} \mathcal{B}$$

$$\begin{array}{ccc} & & FA \\ A & \xrightarrow{\quad} & B \end{array}$$

- $F : \text{Obj}_{\mathcal{A}} \longrightarrow \text{Obj}_{\mathcal{B}}$
- $\text{Hom}_{\mathcal{A}}(A, A') \longrightarrow \text{Hom}_{\mathcal{B}}(FA, FA')$ (respects id and \circ , bijection)
- $\forall B \in \mathcal{B} \quad \exists A \in \mathcal{A} :$

Equivalences

When are two categories “the same” in a categorical way?

$$\mathcal{A} \xrightarrow{F} \mathcal{B}$$

$$\mathcal{A} \quad \rightleftarrows \quad \begin{array}{c} FA \\ \wr \\ B \end{array}$$

- $F : \text{Obj}_{\mathcal{A}} \longrightarrow \text{Obj}_{\mathcal{B}}$
- $\text{Hom}_{\mathcal{A}}(A, A') \longrightarrow \text{Hom}_{\mathcal{B}}(FA, FA')$ (respects id and \circ , bijection)
- $\forall B \in \mathcal{B} \quad \exists A \in \mathcal{A} :$

Equivalences

When are two categories “the same” in a categorical way?

$$\mathcal{A} \xrightarrow{F} \mathcal{B}$$

$$A \quad \rightleftarrows \quad \begin{array}{c} FA \\ \wr \\ B \end{array}$$

- $F : \text{Obj}_{\mathcal{A}} \longrightarrow \text{Obj}_{\mathcal{B}}$
- $\text{Hom}_{\mathcal{A}}(A, A') \longrightarrow \text{Hom}_{\mathcal{B}}(FA, FA')$ (respects id and \circ , bijection)
- $\forall B \in \mathcal{B} \quad \exists A \in \mathcal{A} : \quad FA \simeq B$

Equivalences

When are two categories “the same” in a categorical way?

$$\mathcal{A} \xrightarrow{F} \mathcal{B}$$

$$\mathcal{A} \quad \rightleftarrows \quad \begin{array}{c} FA \\ \wr \\ B \end{array}$$

- $F : \text{Obj}_{\mathcal{A}} \longrightarrow \text{Obj}_{\mathcal{B}}$
- $\text{Hom}_{\mathcal{A}}(A, A') \longrightarrow \text{Hom}_{\mathcal{B}}(FA, FA')$ (respects id and \circ , bijection)
- $\forall B \in \mathcal{B} \quad \exists A \in \mathcal{A} : \quad FA \simeq B$ (essentially surjective)

Equivalences

Let's compare $\text{mat}_{\mathbb{Q}}$ and $\text{vec}_{\mathbb{Q}}$.

Equivalences

Let's compare $\text{mat}_{\mathbb{Q}}$ and $\text{vec}_{\mathbb{Q}}$.

$\text{mat}_{\mathbb{Q}}$

$\text{vec}_{\mathbb{Q}}$

Equivalences

Let's compare $\text{mat}_{\mathbb{Q}}$ and $\text{vec}_{\mathbb{Q}}$.

$\text{mat}_{\mathbb{Q}}$

$\text{vec}_{\mathbb{Q}}$

- $F : \text{Obj}_{\text{mat}_{\mathbb{Q}}} \longrightarrow \text{Obj}_{\text{vec}_{\mathbb{Q}}}$

Equivalences

Let's compare $\text{mat}_{\mathbb{Q}}$ and $\text{vec}_{\mathbb{Q}}$.

$$\text{mat}_{\mathbb{Q}} \xrightarrow{F} \text{vec}_{\mathbb{Q}}$$

- $F : \text{Obj}_{\text{mat}_{\mathbb{Q}}} \longrightarrow \text{Obj}_{\text{vec}_{\mathbb{Q}}}$

Equivalences

Let's compare $\text{mat}_{\mathbb{Q}}$ and $\text{vec}_{\mathbb{Q}}$.

$$\text{mat}_{\mathbb{Q}} \xrightarrow{F} \text{vec}_{\mathbb{Q}}$$

m

- $F : \text{Obj}_{\text{mat}_{\mathbb{Q}}} \longrightarrow \text{Obj}_{\text{vec}_{\mathbb{Q}}}$

Equivalences

Let's compare $\text{mat}_{\mathbb{Q}}$ and $\text{vec}_{\mathbb{Q}}$.

$$\text{mat}_{\mathbb{Q}} \xrightarrow{F} \text{vec}_{\mathbb{Q}}$$

$$m \longmapsto \mathbb{Q}^{1 \times m}$$

- $F : \text{Obj}_{\text{mat}_{\mathbb{Q}}} \longrightarrow \text{Obj}_{\text{vec}_{\mathbb{Q}}}$

Equivalences

Let's compare $\text{mat}_{\mathbb{Q}}$ and $\text{vec}_{\mathbb{Q}}$.

$$\text{mat}_{\mathbb{Q}} \xrightarrow{F} \text{vec}_{\mathbb{Q}}$$

$$m \longmapsto \mathbb{Q}^{1 \times m}$$

- $F : \text{Obj}_{\text{mat}_{\mathbb{Q}}} \longrightarrow \text{Obj}_{\text{vec}_{\mathbb{Q}}}$
- $\text{Hom}_{\text{mat}_{\mathbb{Q}}}(m, n) \longrightarrow \text{Hom}_{\text{vec}_{\mathbb{Q}}}(\mathbb{Q}^{1 \times m}, \mathbb{Q}^{1 \times n})$

Equivalences

Let's compare $\text{mat}_{\mathbb{Q}}$ and $\text{vec}_{\mathbb{Q}}$.

$$\text{mat}_{\mathbb{Q}} \xrightarrow{F} \text{vec}_{\mathbb{Q}}$$

$$\begin{array}{c} m \\ (a_{ij})_{ij} \downarrow \\ n \end{array}$$

- $F : \text{Obj}_{\text{mat}_{\mathbb{Q}}} \longrightarrow \text{Obj}_{\text{vec}_{\mathbb{Q}}}$
- $\text{Hom}_{\text{mat}_{\mathbb{Q}}}(m, n) \longrightarrow \text{Hom}_{\text{vec}_{\mathbb{Q}}}(\mathbb{Q}^{1 \times m}, \mathbb{Q}^{1 \times n})$

Equivalences

Let's compare $\text{mat}_{\mathbb{Q}}$ and $\text{vec}_{\mathbb{Q}}$.

$$\text{mat}_{\mathbb{Q}} \xrightarrow{F} \text{vec}_{\mathbb{Q}}$$

$$\begin{array}{ccc} \begin{array}{c} m \\ \downarrow \\ (a_{ij})_{ij} \\ \downarrow \\ n \end{array} & \longmapsto & \begin{array}{c} \mathbb{Q}^{1 \times m} \\ \downarrow v \mapsto v \cdot (a_{ij})_{ij} \\ \mathbb{Q}^{1 \times n} \end{array} \end{array}$$

- $F : \text{Obj}_{\text{mat}_{\mathbb{Q}}} \longrightarrow \text{Obj}_{\text{vec}_{\mathbb{Q}}}$
- $\text{Hom}_{\text{mat}_{\mathbb{Q}}}(m, n) \longrightarrow \text{Hom}_{\text{vec}_{\mathbb{Q}}}(\mathbb{Q}^{1 \times m}, \mathbb{Q}^{1 \times n})$

Equivalences

Let's compare $\text{mat}_{\mathbb{Q}}$ and $\text{vec}_{\mathbb{Q}}$.

$$\text{mat}_{\mathbb{Q}} \xrightarrow{F} \text{vec}_{\mathbb{Q}}$$

$$\begin{array}{ccc} \begin{array}{c} m \\ \downarrow \\ (a_{ij})_{ij} \\ \downarrow \\ n \end{array} & \longmapsto & \begin{array}{c} \mathbb{Q}^{1 \times m} \\ \downarrow v \mapsto v \cdot (a_{ij})_{ij} \\ \mathbb{Q}^{1 \times n} \end{array} \end{array}$$

- $F : \text{Obj}_{\text{mat}_{\mathbb{Q}}} \longrightarrow \text{Obj}_{\text{vec}_{\mathbb{Q}}}$
- $\text{Hom}_{\text{mat}_{\mathbb{Q}}}(m, n) \longrightarrow \text{Hom}_{\text{vec}_{\mathbb{Q}}}(\mathbb{Q}^{1 \times m}, \mathbb{Q}^{1 \times n})$ (respects id and \circ)

Equivalences

Let's compare $\text{mat}_{\mathbb{Q}}$ and $\text{vec}_{\mathbb{Q}}$.

$$\text{mat}_{\mathbb{Q}} \xrightarrow{F} \text{vec}_{\mathbb{Q}}$$

$$\begin{array}{ccc} \begin{array}{c} m \\ \downarrow \\ (a_{ij})_{ij} \\ \downarrow \\ n \end{array} & \longmapsto & \begin{array}{c} \mathbb{Q}^{1 \times m} \\ \downarrow v \mapsto v \cdot (a_{ij})_{ij} \\ \mathbb{Q}^{1 \times n} \end{array} \end{array}$$

- $F : \text{Obj}_{\text{mat}_{\mathbb{Q}}} \longrightarrow \text{Obj}_{\text{vec}_{\mathbb{Q}}}$
- $\text{Hom}_{\text{mat}_{\mathbb{Q}}}(m, n) \longrightarrow \text{Hom}_{\text{vec}_{\mathbb{Q}}}(\mathbb{Q}^{1 \times m}, \mathbb{Q}^{1 \times n})$ (respects id and \circ)
- $\forall B \in \text{vec}_{\mathbb{Q}}$

Equivalences

Let's compare $\text{mat}_{\mathbb{Q}}$ and $\text{vec}_{\mathbb{Q}}$.

$$\text{mat}_{\mathbb{Q}} \xrightarrow{F} \text{vec}_{\mathbb{Q}}$$

- $F : \text{Obj}_{\text{mat}_{\mathbb{Q}}} \longrightarrow \text{Obj}_{\text{vec}_{\mathbb{Q}}}$
- $\text{Hom}_{\text{mat}_{\mathbb{Q}}}(m, n) \longrightarrow \text{Hom}_{\text{vec}_{\mathbb{Q}}}(\mathbb{Q}^{1 \times m}, \mathbb{Q}^{1 \times n})$ (respects id and \circ)
- $\forall B \in \text{vec}_{\mathbb{Q}}$

Equivalences

Let's compare $\text{mat}_{\mathbb{Q}}$ and $\text{vec}_{\mathbb{Q}}$.

$$\text{mat}_{\mathbb{Q}} \xrightarrow{F} \text{vec}_{\mathbb{Q}}$$

B

- $F : \text{Obj}_{\text{mat}_{\mathbb{Q}}} \longrightarrow \text{Obj}_{\text{vec}_{\mathbb{Q}}}$
- $\text{Hom}_{\text{mat}_{\mathbb{Q}}}(m, n) \longrightarrow \text{Hom}_{\text{vec}_{\mathbb{Q}}}(\mathbb{Q}^{1 \times m}, \mathbb{Q}^{1 \times n})$ (respects id and \circ)
- $\forall B \in \text{vec}_{\mathbb{Q}}$

Equivalences

Let's compare $\text{mat}_{\mathbb{Q}}$ and $\text{vec}_{\mathbb{Q}}$.

$$\text{mat}_{\mathbb{Q}} \xrightarrow{F} \text{vec}_{\mathbb{Q}}$$

B

- $F : \text{Obj}_{\text{mat}_{\mathbb{Q}}} \longrightarrow \text{Obj}_{\text{vec}_{\mathbb{Q}}}$
- $\text{Hom}_{\text{mat}_{\mathbb{Q}}}(m, n) \longrightarrow \text{Hom}_{\text{vec}_{\mathbb{Q}}}(\mathbb{Q}^{1 \times m}, \mathbb{Q}^{1 \times n})$ (respects id and \circ)
- $\forall B \in \text{vec}_{\mathbb{Q}} \quad \exists m \in \text{mat}_{\mathbb{Q}} :$

Equivalences

Let's compare $\text{mat}_{\mathbb{Q}}$ and $\text{vec}_{\mathbb{Q}}$.

$$\text{mat}_{\mathbb{Q}} \xrightarrow{F} \text{vec}_{\mathbb{Q}}$$

m

B

- $F : \text{Obj}_{\text{mat}_{\mathbb{Q}}} \longrightarrow \text{Obj}_{\text{vec}_{\mathbb{Q}}}$
- $\text{Hom}_{\text{mat}_{\mathbb{Q}}}(m, n) \longrightarrow \text{Hom}_{\text{vec}_{\mathbb{Q}}}(\mathbb{Q}^{1 \times m}, \mathbb{Q}^{1 \times n})$ (respects id and \circ)
- $\forall B \in \text{vec}_{\mathbb{Q}} \quad \exists m \in \text{mat}_{\mathbb{Q}} :$

Equivalences

Let's compare $\text{mat}_{\mathbb{Q}}$ and $\text{vec}_{\mathbb{Q}}$.

$$\text{mat}_{\mathbb{Q}} \xrightarrow{F} \text{vec}_{\mathbb{Q}}$$

$$m \quad \xleftrightarrow{\quad} \quad \begin{matrix} \mathbb{Q}^{1 \times m} \\ B \end{matrix}$$

- $F : \text{Obj}_{\text{mat}_{\mathbb{Q}}} \longrightarrow \text{Obj}_{\text{vec}_{\mathbb{Q}}}$
- $\text{Hom}_{\text{mat}_{\mathbb{Q}}}(m, n) \longrightarrow \text{Hom}_{\text{vec}_{\mathbb{Q}}}(\mathbb{Q}^{1 \times m}, \mathbb{Q}^{1 \times n})$ (respects id and \circ)
- $\forall B \in \text{vec}_{\mathbb{Q}} \quad \exists m \in \text{mat}_{\mathbb{Q}} :$

Equivalences

Let's compare $\text{mat}_{\mathbb{Q}}$ and $\text{vec}_{\mathbb{Q}}$.

$$\text{mat}_{\mathbb{Q}} \xrightarrow{F} \text{vec}_{\mathbb{Q}}$$

$$m \quad \mapsto \quad \begin{array}{c} \mathbb{Q}^{1 \times m} \\ \wr \\ B \end{array}$$

- $F : \text{Obj}_{\text{mat}_{\mathbb{Q}}} \longrightarrow \text{Obj}_{\text{vec}_{\mathbb{Q}}}$
- $\text{Hom}_{\text{mat}_{\mathbb{Q}}}(m, n) \longrightarrow \text{Hom}_{\text{vec}_{\mathbb{Q}}}(\mathbb{Q}^{1 \times m}, \mathbb{Q}^{1 \times n})$ (respects id and \circ)
- $\forall B \in \text{vec}_{\mathbb{Q}} \quad \exists m \in \text{mat}_{\mathbb{Q}} :$

Equivalences

Let's compare $\text{mat}_{\mathbb{Q}}$ and $\text{vec}_{\mathbb{Q}}$.

$$\text{mat}_{\mathbb{Q}} \xrightarrow{F} \text{vec}_{\mathbb{Q}}$$

$$m \quad \mapsto \quad \begin{array}{c} \mathbb{Q}^{1 \times m} \\ \wr \\ B \end{array}$$

- $F : \text{Obj}_{\text{mat}_{\mathbb{Q}}} \longrightarrow \text{Obj}_{\text{vec}_{\mathbb{Q}}}$
- $\text{Hom}_{\text{mat}_{\mathbb{Q}}}(m, n) \longrightarrow \text{Hom}_{\text{vec}_{\mathbb{Q}}}(\mathbb{Q}^{1 \times m}, \mathbb{Q}^{1 \times n})$ (respects id and \circ)
- $\forall B \in \text{vec}_{\mathbb{Q}} \quad \exists m \in \text{mat}_{\mathbb{Q}} : \quad \mathbb{Q}^{1 \times m} \simeq B$

Equivalences

Let's compare $\text{mat}_{\mathbb{Q}}$ and $\text{vec}_{\mathbb{Q}}$.

$$\text{mat}_{\mathbb{Q}} \xrightarrow{F} \text{vec}_{\mathbb{Q}}$$

$$m \quad \mapsto \quad \begin{array}{c} \mathbb{Q}^{1 \times m} \\ \wr \\ B \end{array}$$

- $F : \text{Obj}_{\text{mat}_{\mathbb{Q}}} \longrightarrow \text{Obj}_{\text{vec}_{\mathbb{Q}}}$
- $\text{Hom}_{\text{mat}_{\mathbb{Q}}}(m, n) \longrightarrow \text{Hom}_{\text{vec}_{\mathbb{Q}}}(\mathbb{Q}^{1 \times m}, \mathbb{Q}^{1 \times n})$ (respects id and \circ)
- $\forall B \in \text{vec}_{\mathbb{Q}} \quad \exists m \in \text{mat}_{\mathbb{Q}} : \quad \mathbb{Q}^{1 \times m} \simeq B \quad m = \dim(B)$

$$\mathbf{mat}_{\mathbb{Q}} \simeq \mathbf{vec}_{\mathbb{Q}}$$

Categorical abstraction

$$\mathbf{mat}_{\mathbb{Q}} \simeq \mathbf{vec}_{\mathbb{Q}}$$

For a category theorist, these two categories look the same.

Q: What is a finite dimensional \mathbb{Q} -vector space?

Categorical abstraction

Q: What is a finite dimensional \mathbb{Q} -vector space?

A: An object in the **category** of finite dim. \mathbb{Q} -vector spaces.

Categorical abstraction

Q: What is a finite dimensional \mathbb{Q} -vector space?

A: An object in the **category** of finite dim. \mathbb{Q} -vector spaces.

Q: What is the **category** of finite dimensional \mathbb{Q} -vector spaces?

Categorical abstraction

Q: What is a finite dimensional \mathbb{Q} -vector space?

A: An object in the **category** of finite dim. \mathbb{Q} -vector spaces.

Q: What is the **category** of finite dimensional \mathbb{Q} -vector spaces?

$$\mathbf{mat}_{\mathbb{Q}} \simeq \mathbf{vec}_{\mathbb{Q}}$$

Categorical abstraction

Q: What is a finite dimensional \mathbb{Q} -vector space?

A: An object in the **category** of finite dim. \mathbb{Q} -vector spaces.

Q: What is the **category** of finite dimensional \mathbb{Q} -vector spaces?

$$\text{mat}_{\mathbb{Q}} \simeq \text{vec}_{\mathbb{Q}}$$

$\text{mat}_{\mathbb{Q}}$ is a **computerfriendly** model of $\text{vec}_{\mathbb{Q}}$.

Categorical abstraction

Q: What is a finite dimensional \mathbb{Q} -vector space?

A: An object in the **category** of finite dim. \mathbb{Q} -vector spaces.

Q: What is the **category** of finite dimensional \mathbb{Q} -vector spaces?

$$\text{mat}_{\mathbb{Q}} \simeq \text{vec}_{\mathbb{Q}}$$

$\text{mat}_{\mathbb{Q}}$ is a **computerfriendly** model of $\text{vec}_{\mathbb{Q}}$.

We want to use categories to model **computational contexts** instead of “isolated” objects.

Categorical abstraction is a powerful organizing principle and computational tool.

- 1 What is categorical abstraction?
- 2 How can it be used as an organizing principle?
- 3 Why is it a computational tool?

Computable categories

A category becomes computable through

Computable categories

A category becomes computable through

- Data structures for *objects* and *morphisms*

Computable categories

A category becomes computable through

- Data structures for *objects* and *morphisms*
- Algorithms to compute the *composition* of morphisms

Computable categories

A category becomes computable through

- Data structures for *objects* and *morphisms*
- Algorithms to compute the *composition* of morphisms and *identity morphisms* of objects

Computable categories

A category becomes computable through

- Data structures for *objects* and *morphisms*
- Algorithms to compute the *composition* of morphisms and *identity morphisms* of objects

Computable categories

A category becomes computable through

- Data structures for *objects* and *morphisms*
- Algorithms to compute the *composition* of morphisms and *identity morphisms* of objects

$\text{mat}_{\mathbb{Q}}$

Computable categories

A category becomes computable through

- Data structures for *objects* and *morphisms*
- Algorithms to compute the *composition* of morphisms and *identity morphisms* of objects

$\text{mat}_{\mathbb{Q}}$

1

2

1

Computable categories

A category becomes computable through

- Data structures for *objects* and *morphisms*
- Algorithms to compute the *composition* of morphisms and *identity morphisms* of objects

$\text{mat}_{\mathbb{Q}}$

1

2

1

Computable categories

A category becomes computable through

- Data structures for *objects* and *morphisms*
- Algorithms to compute the *composition* of morphisms and *identity morphisms* of objects

$\text{mat}_{\mathbb{Q}}$

$$1 \xrightarrow{\begin{pmatrix} 1 & 2 \end{pmatrix}} 2 \xrightarrow{\begin{pmatrix} 3 \\ 4 \end{pmatrix}} 1$$

Computable categories

A category becomes computable through

- Data structures for *objects* and *morphisms*
- Algorithms to compute the *composition* of morphisms and *identity morphisms* of objects

$\text{mat}_{\mathbb{Q}}$

$$1 \xrightarrow{\begin{pmatrix} 1 & 2 \end{pmatrix}} 2 \xrightarrow{\begin{pmatrix} 3 \\ 4 \end{pmatrix}} 1$$

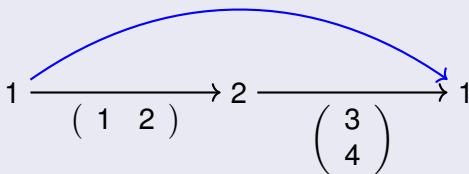
Computable categories

A category becomes computable through

- Data structures for *objects* and *morphisms*
- Algorithms to compute the *composition* of morphisms and *identity morphisms* of objects

$\text{mat}_{\mathbb{Q}}$

$$\begin{pmatrix} 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 4 \end{pmatrix} = (11)$$



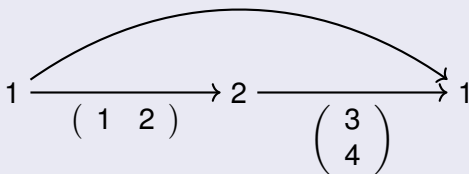
Computable categories

A category becomes computable through

- Data structures for *objects* and *morphisms*
- Algorithms to compute the *composition* of morphisms and *identity morphisms* of objects

$\text{mat}_{\mathbb{Q}}$

$$\begin{pmatrix} 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 4 \end{pmatrix} = (11)$$



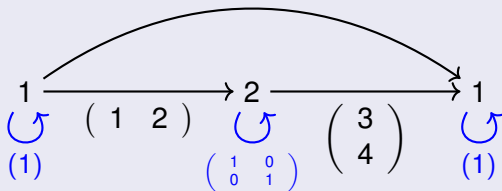
Computable categories

A category becomes computable through

- Data structures for *objects* and *morphisms*
- Algorithms to compute the *composition* of morphisms and *identity morphisms* of objects

$\text{mat}_{\mathbb{Q}}$

$$\begin{pmatrix} 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 4 \end{pmatrix} = (11)$$



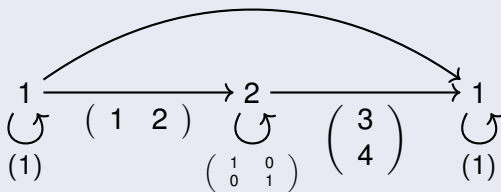
Computable categories

A category becomes computable through

- Data structures for *objects* and *morphisms*
- Algorithms to compute the *composition* of morphisms and *identity morphisms* of objects

$\text{mat}_{\mathbb{Q}}$

$$\begin{pmatrix} 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 4 \end{pmatrix} = (11)$$



The language of category theory

$\text{vec}_{\mathbb{Q}}$ and $\text{mat}_{\mathbb{Q}}$ are examples of abelian categories.

The language of category theory

$\text{vec}_{\mathbb{Q}}$ and $\text{mat}_{\mathbb{Q}}$ are examples of abelian categories.

Some categorical operations in abelian categories

The language of category theory

$\text{vec}_{\mathbb{Q}}$ and $\text{mat}_{\mathbb{Q}}$ are examples of abelian categories.

Some categorical operations in abelian categories

- $\oplus : \text{Obj} \times \text{Obj} \rightarrow \text{Obj}$

The language of category theory

$\text{vec}_{\mathbb{Q}}$ and $\text{mat}_{\mathbb{Q}}$ are examples of abelian categories.

Some categorical operations in abelian categories

- $\oplus : \text{Obj} \times \text{Obj} \rightarrow \text{Obj}$
- $+, - : \text{Hom}(A, B) \times \text{Hom}(A, B) \rightarrow \text{Hom}(A, B)$

The language of category theory

$\text{vec}_{\mathbb{Q}}$ and $\text{mat}_{\mathbb{Q}}$ are examples of abelian categories.

Some categorical operations in abelian categories

- $\oplus : \text{Obj} \times \text{Obj} \rightarrow \text{Obj}$
- $+, - : \text{Hom}(A, B) \times \text{Hom}(A, B) \rightarrow \text{Hom}(A, B)$
- $\ker : \text{Hom}(A, B) \rightarrow \text{Obj}$

The language of category theory

$\text{vec}_{\mathbb{Q}}$ and $\text{mat}_{\mathbb{Q}}$ are examples of abelian categories.

Some categorical operations in abelian categories

- $\oplus : \text{Obj} \times \text{Obj} \rightarrow \text{Obj}$
- $+, - : \text{Hom}(A, B) \times \text{Hom}(A, B) \rightarrow \text{Hom}(A, B)$
- $\ker : \text{Hom}(A, B) \rightarrow \text{Obj}$

Categorical description of the kernel

What do we want from a **kernel**?

Categorical description of the kernel

What do we want from a **kernel**?

Given $\alpha : V \longrightarrow W$ in $\text{vec}_{\mathbb{Q}}$.

Categorical description of the kernel

What do we want from a **kernel**?

Given $\alpha : V \longrightarrow W$ in $\text{vec}_{\mathbb{Q}}$.

$$\ker(\alpha) = \{v \in V \mid \alpha(v) = 0\}$$

Categorical description of the kernel

What do we want from a **kernel**?

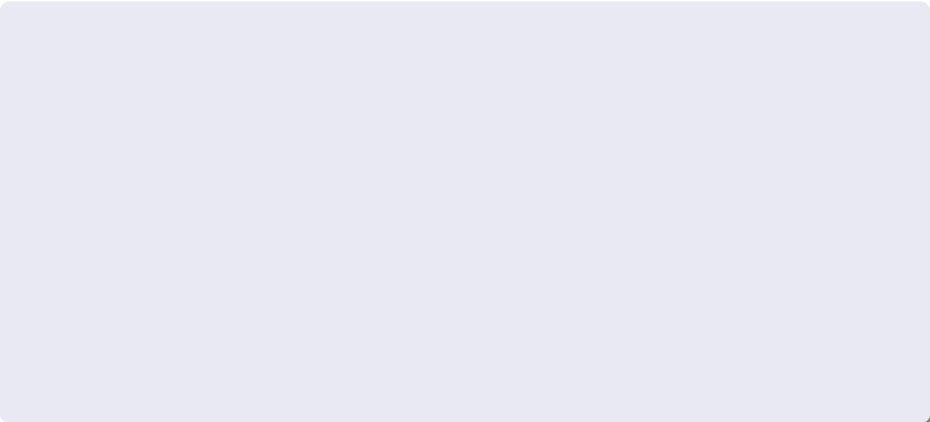
Given $\alpha : V \longrightarrow W$ in $\mathbf{vec}_{\mathbb{Q}}$.

$$\ker(\alpha) = \{v \in V \mid \alpha(v) = 0\}$$

Complete understanding about what is mapped to 0.

Categorical description of the kernel

Complete understanding about what is mapped to 0.



Categorical description of the kernel

Complete understanding about what is mapped to 0.

$$m \xrightarrow{(a_{ij})_{ij}} n$$

Categorical description of the kernel

Complete understanding about what is mapped to 0.

$$\dim(\ker(a_{ij}))$$

$$m \xrightarrow{(a_{ij})_{ij}} n$$

Categorical description of the kernel

Complete understanding about what is mapped to 0.

$$\dim(\ker(a_{ij}))$$

$$3 \xrightarrow{\begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix}} 2$$

Categorical description of the kernel

Complete understanding about what is mapped to 0.

2

$$3 \xrightarrow{\begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix}} 2$$

Categorical description of the kernel

Complete understanding about what is mapped to 0.

$$\dim(\ker(a_{ij}))$$

$$3 \xrightarrow{\begin{pmatrix} 1 & -1 \\ -1 & 1 \\ 1 & -1 \end{pmatrix}} 2$$

Categorical description of the kernel

Complete understanding about what is mapped to 0.

2

$$3 \xrightarrow{\begin{pmatrix} 1 & -1 \\ -1 & 1 \\ 1 & -1 \end{pmatrix}} 2$$

Categorical description of the kernel

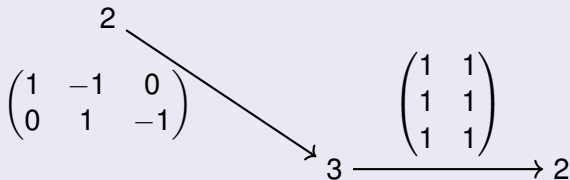
Complete understanding about what is mapped to 0.

2

$$3 \xrightarrow{\begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix}} 2$$

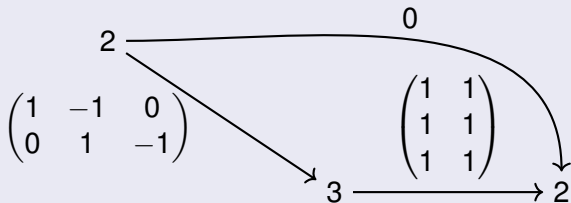
Categorical description of the kernel

Complete understanding about what is mapped to 0.



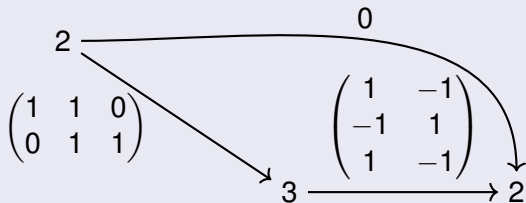
Categorical description of the kernel

Complete understanding about what is mapped to 0.



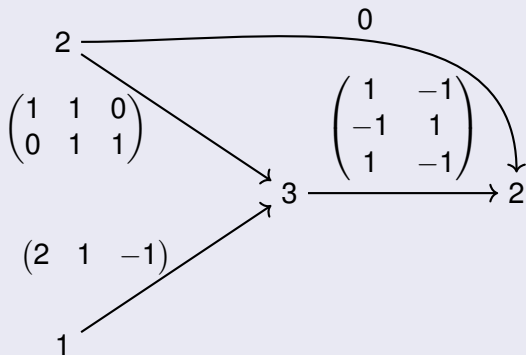
Categorical description of the kernel

Complete understanding about what is mapped to 0.



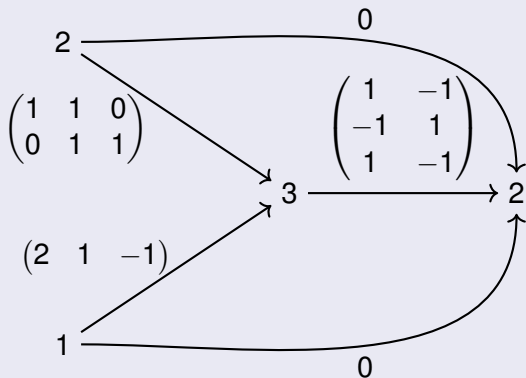
Categorical description of the kernel

Complete understanding about what is mapped to 0.



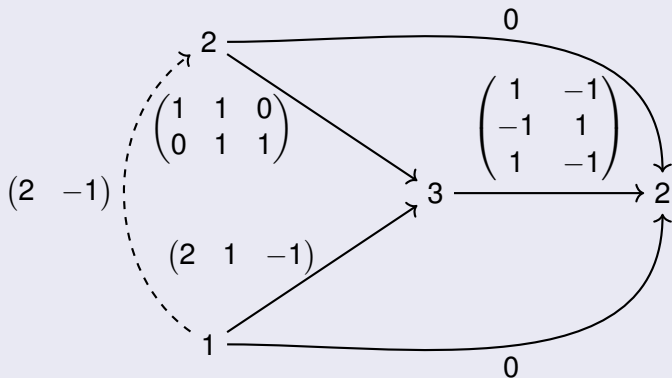
Categorical description of the kernel

Complete understanding about what is mapped to 0.



Categorical description of the kernel

Complete understanding about what is mapped to 0.



Categorical description of the kernel

Let $\varphi \in \text{Hom}(A, B)$.

Categorical description of the kernel

Let $\varphi \in \text{Hom}(A, B)$.

$$A \xrightarrow{\varphi} B$$

Categorical description of the kernel

Let $\varphi \in \text{Hom}(A, B)$. To fully describe the kernel of $\varphi \dots$

$$A \xrightarrow{\varphi} B$$

Categorical description of the kernel

Let $\varphi \in \text{Hom}(A, B)$. To fully describe the kernel of $\varphi \dots$

\dots one needs an object $\text{KernelObject}(\varphi)$,

$\text{KernelObject}(\varphi)$

$$A \xrightarrow{\varphi} B$$

Categorical description of the kernel

Let $\varphi \in \text{Hom}(A, B)$. To fully describe the kernel of $\varphi \dots$

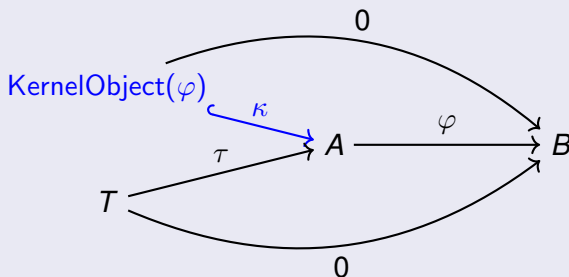
\dots one needs an object $\text{KernelObject}(\varphi)$,
its embedding $\kappa = \text{KernelEmbedding}(\varphi)$,

$$\text{KernelObject}(\varphi) \xrightarrow{\kappa} A \xrightarrow{\varphi} B$$

Categorical description of the kernel

Let $\varphi \in \text{Hom}(A, B)$. To fully describe the kernel of $\varphi \dots$

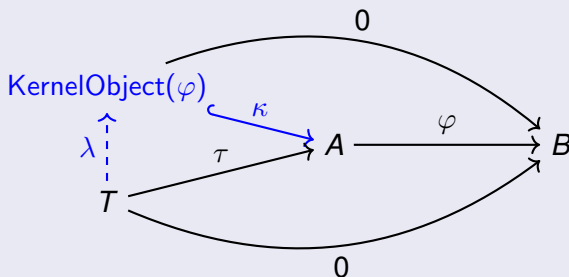
\dots one needs an object $\text{KernelObject}(\varphi)$,
its embedding $\kappa = \text{KernelEmbedding}(\varphi)$,
and for every test morphism τ



Categorical description of the kernel

Let $\varphi \in \text{Hom}(A, B)$. To fully describe the kernel of $\varphi \dots$

\dots one needs an object $\text{KernelObject}(\varphi)$,
its embedding $\kappa = \text{KernelEmbedding}(\varphi)$,
and for every test morphism τ
a *unique* morphism $\lambda = \text{KernelLift}(\varphi, \tau)$



Implementation of the kernel

$\text{mat}_{\mathbb{Q}}$

$$\text{Obj} := \mathbb{N}_0, \quad \text{Hom}(m, n) := \mathbb{Q}^{m \times n}$$

Implementation of the kernel

$\text{mat}_{\mathbb{Q}}$

$$\text{Obj} := \mathbb{N}_0, \quad \text{Hom}(m, n) := \mathbb{Q}^{m \times n}$$

$$m \xrightarrow{(a_{ij})_{ij}} n$$

Implementation of the kernel

$\text{mat}_{\mathbb{Q}}$

$$\text{Obj} := \mathbb{N}_0, \quad \text{Hom}(m, n) := \mathbb{Q}^{m \times n}$$

$\text{KernelObject}((a_{ij})_{ij})$

$$m \xrightarrow{(a_{ij})_{ij}} n$$

Implementation of the kernel

$\text{mat}_{\mathbb{Q}}$

$$\text{Obj} := \mathbb{N}_0, \quad \text{Hom}(m, n) := \mathbb{Q}^{m \times n}$$

$\text{KernelObject}((a_{ij})_{ij})$

$$m \xrightarrow{(a_{ij})_{ij}} n$$

Compute

- $\text{KernelObject}((a_{ij})_{ij}) := m - \text{rank}((a_{ij})_{ij})$

Implementation of the kernel

$\text{mat}_{\mathbb{Q}}$

$$\text{Obj} := \mathbb{N}_0, \quad \text{Hom}(m, n) := \mathbb{Q}^{m \times n}$$

$$\text{KernelObject}((a_{ij})_{ij}) \xrightarrow{\kappa} m \xrightarrow{(a_{ij})_{ij}} n$$

Compute

- $\text{KernelObject}((a_{ij})_{ij}) := m - \text{rank}((a_{ij})_{ij})$

Implementation of the kernel

$\text{mat}_{\mathbb{Q}}$

$$\text{Obj} := \mathbb{N}_0, \quad \text{Hom}(m, n) := \mathbb{Q}^{m \times n}$$

$$\text{KernelObject}((a_{ij})_{ij}) \xrightarrow{\kappa} m \xrightarrow{(a_{ij})_{ij}} n$$

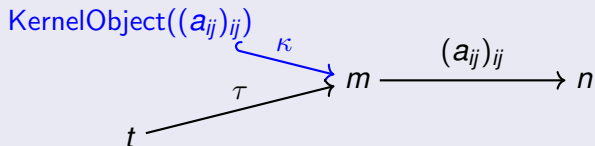
Compute

- $\text{KernelObject}((a_{ij})_{ij}) := m - \text{rank}((a_{ij})_{ij})$
- $\kappa :=$ matrix whose rows form a basis of solutions of $X \cdot (a_{ij})_{ij} = 0$

Implementation of the kernel

$\text{mat}_{\mathbb{Q}}$

$$\text{Obj} := \mathbb{N}_0, \quad \text{Hom}(m, n) := \mathbb{Q}^{m \times n}$$



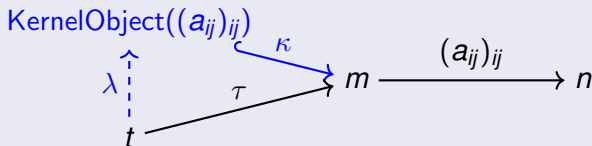
Compute

- $\text{KernelObject}((a_{ij})_{ij}) := m - \text{rank}((a_{ij})_{ij})$
- $\kappa :=$ matrix whose rows form a basis of solutions of $X \cdot (a_{ij})_{ij} = 0$

Implementation of the kernel

$\text{mat}_{\mathbb{Q}}$

$$\text{Obj} := \mathbb{N}_0, \quad \text{Hom}(m, n) := \mathbb{Q}^{m \times n}$$



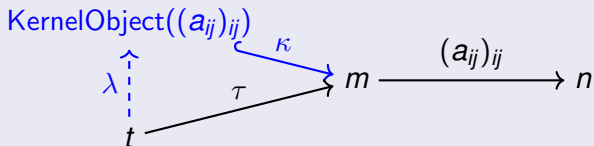
Compute

- $\text{KernelObject}((a_{ij})_{ij}) := m - \text{rank}((a_{ij})_{ij})$
- $\kappa :=$ matrix whose rows form a basis of solutions of $X \cdot (a_{ij})_{ij} = 0$

Implementation of the kernel

$\text{mat}_{\mathbb{Q}}$

$$\text{Obj} := \mathbb{N}_0, \quad \text{Hom}(m, n) := \mathbb{Q}^{m \times n}$$



Compute

- $\text{KernelObject}((a_{ij})_{ij}) := m - \text{rank}((a_{ij})_{ij})$
- $\kappa :=$ matrix whose rows form a basis of solutions of $X \cdot (a_{ij})_{ij} = 0$
- $\lambda :=$ the unique solution of $X \cdot \kappa = \tau$

The language of category theory

The language of category theory

Given a diagram of vector spaces:

The language of category theory

Given a diagram of vector spaces:

$$\begin{array}{ccccc} \text{ker} & \hookrightarrow & A' & \xrightarrow{\quad} & B' \\ & & \downarrow \alpha & & \downarrow \\ \text{ker} & \hookrightarrow & A & \xrightarrow{\quad} & B \end{array}$$

The language of category theory

Given a diagram of vector spaces:

$$\begin{array}{ccccc} \text{ker} & \hookrightarrow & A' & \xrightarrow{\quad} & B' \\ \downarrow & & \downarrow \alpha & & \downarrow \\ \text{ker} & \hookrightarrow & A & \xrightarrow{\quad} & B \end{array}$$

The language of category theory

Given a diagram of vector spaces:

$$\begin{array}{ccccc} x \in \text{ker} & \hookrightarrow & A' & \xrightarrow{\quad} & B' \\ \downarrow & & \downarrow \alpha & & \downarrow \\ \text{ker} & \hookrightarrow & A & \xrightarrow{\quad} & B \end{array}$$

The language of category theory

Given a diagram of vector spaces:

$$\begin{array}{ccccc} x \in \text{ker} & \hookrightarrow & x \in A' & \xrightarrow{\quad} & B' \\ \downarrow & & \downarrow \alpha & & \downarrow \\ \text{ker} & \hookrightarrow & A & \xrightarrow{\quad} & B \end{array}$$

The language of category theory

Given a diagram of vector spaces:

$$\begin{array}{ccccc} x \in \text{ker} & \hookrightarrow & x \in A' & \xrightarrow{\quad} & B' \\ \downarrow & & \downarrow \alpha & & \downarrow \\ \text{ker} & \hookrightarrow & \alpha(x) \in A & \xrightarrow{\quad} & B \end{array}$$

The language of category theory

Given a diagram of vector spaces:

$$\begin{array}{ccccc} x \in \text{ker} & \hookrightarrow & x \in A' & \xrightarrow{\quad} & B' \\ \downarrow & & \downarrow \alpha & & \downarrow \\ \text{ker} & \hookrightarrow & \alpha(x) \in A & \xrightarrow{\quad} & 0 \in B \end{array}$$

The language of category theory

Given a diagram of vector spaces:

$$\begin{array}{ccccc} x \in \text{ker} & \hookrightarrow & x \in A' & \xrightarrow{\quad} & B' \\ \downarrow & & \downarrow \alpha & & \downarrow \\ \alpha(x) \in \text{ker} & \hookrightarrow & \alpha(x) \in A & \xrightarrow{\quad} & 0 \in B \end{array}$$

The language of category theory

Given a diagram of vector spaces:

$$\begin{array}{ccccc} x \in \text{ker} & \hookrightarrow & x \in A' & \xrightarrow{\quad} & B' \\ \downarrow & & \downarrow \alpha & & \downarrow \\ \alpha(x) \in \text{ker} & \hookrightarrow & \alpha(x) \in A & \xrightarrow{\quad} & 0 \in B \end{array}$$

The language of category theory

The same example in the language of category theory:

$$\begin{array}{ccccc} \text{ker} & \xrightarrow{\kappa'} & A' & \xrightarrow{\quad} & B' \\ \downarrow & & \downarrow \alpha & & \downarrow \\ \text{ker} & \xrightarrow{\quad} & A & \xrightarrow{\varphi} & B \end{array}$$

The language of category theory

The same example in the language of category theory:

$$\begin{array}{ccccc} \text{ker} & \xrightarrow{\kappa'} & A' & \xrightarrow{\quad} & B' \\ \downarrow & & \downarrow \alpha & & \downarrow \\ \text{ker} & \xrightarrow{\quad} & A & \xrightarrow{\varphi} & B \end{array}$$

\downarrow

The language of category theory

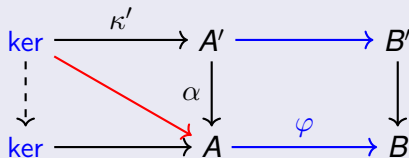
The same example in the language of category theory:

$$\begin{array}{ccccc} \text{ker} & \xrightarrow{\kappa'} & A' & \xrightarrow{\quad} & B' \\ \downarrow & & \downarrow \alpha & & \downarrow \\ \text{ker} & \xrightarrow{\quad} & A & \xrightarrow{\varphi} & B \end{array}$$

$$\downarrow =$$

The language of category theory

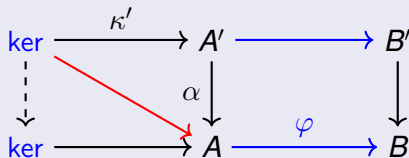
The same example in the language of category theory:



$$\downarrow = \alpha \circ \kappa'$$

The language of category theory

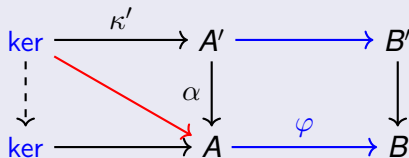
The same example in the language of category theory:



$$\downarrow = \text{KernelLift}(\varphi, \alpha \circ \kappa')$$

The language of category theory

The same example in the language of category theory:



$$\downarrow = \text{KernelLift}(\varphi, \alpha \circ \kappa')$$

This term may be interpreted in other contexts as well.

The language of category theory

The language of category theory

$\text{vec}_{\mathbb{Q}}$

The language of category theory

$\text{vec}_{\mathbb{Q}}$


$\text{mat}_{\mathbb{Q}}$

The language of category theory


$$\mathbf{vec}_{\mathbb{Q}} \quad \simeq \quad \mathbf{mat}_{\mathbb{Q}}$$

The language of category theory

$$\mathbf{vec}_{\mathbb{Q}} \quad \simeq \quad \mathbf{mat}_{\mathbb{Q}}$$


categorical abstraction

The language of category theory

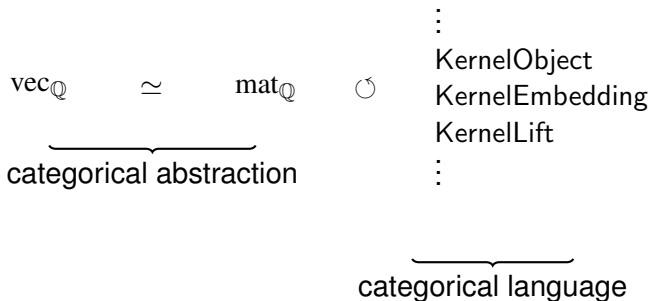
$\text{vec}_{\mathbb{Q}} \quad \simeq \quad \text{mat}_{\mathbb{Q}}$

categorical abstraction

\vdots
KernelObject
KernelEmbedding
KernelLift
 \vdots

The language of category theory

$$\begin{array}{c} \text{vec}_{\mathbb{Q}} \quad \simeq \quad \text{mat}_{\mathbb{Q}} \quad \circlearrowright \\ \underbrace{\hspace{10em}} \\ \text{categorical abstraction} \end{array} \quad \begin{array}{l} \vdots \\ \text{KernelObject} \\ \text{KernelEmbedding} \\ \text{KernelLift} \\ \vdots \end{array}$$

The language of category theory



The language of category theory

\mathcal{A} \circlearrowleft $\begin{array}{c} \vdots \\ \text{KernelObject} \\ \text{KernelEmbedding} \\ \text{KernelLift} \\ \vdots \end{array}$

$\underbrace{\hspace{10em}}$
categorical language

An introduction to finitely presented modules

Let R be a ring.

Definition

A (left) R -module M is called **finitely presented** if

$$M \cong \frac{R^{1 \times n}}{\langle r_1, \dots, r_m \rangle}$$

for $n, m \in \mathbb{N}_0$, $r_1, \dots, r_m \in R^{1 \times n}$.

Examples

$$M \cong \frac{R^{1 \times n}}{\langle r_1, \dots, r_m \rangle}$$

Examples

$$M \cong \frac{R^{1 \times n}}{\langle r_1, \dots, r_m \rangle}$$

$$R = \mathbb{Q}[x, y, z]$$

Examples

$$M \cong \frac{R^{1 \times n}}{\langle r_1, \dots, r_m \rangle}$$

$$R = \mathbb{Q}[x, y, z]$$

$$\frac{\mathbb{Q}[x, y, z]}{\langle x^2 + y^2 + z^2 - 1 \rangle}$$

Examples

$$M \cong \frac{R^{1 \times n}}{\langle r_1, \dots, r_m \rangle}$$

$$R = \mathbb{Q}[x, y, z]$$

$$\frac{\mathbb{Q}[x, y, z]}{\langle x^2 + y^2 + z^2 - 1 \rangle}$$

$$\frac{\mathbb{Q}[x, y, z]^{1 \times 2}}{\langle (x \ -y), (y \ -x) \rangle}$$

Examples

$$M \cong \frac{R^{1 \times n}}{\langle r_1, \dots, r_m \rangle}$$

$$R = \mathbb{Q}[x, y, z]$$

$$\frac{\mathbb{Q}[x, y, z]}{\langle x^2 + y^2 + z^2 - 1 \rangle}$$

$$\frac{\mathbb{Q}[x, y, z]^{1 \times 2}}{\langle (x \ -y), (y \ -x) \rangle}$$

$$\frac{\mathbb{Q}[x, y, z]^{1 \times 6}}{\langle \text{Rows of } \begin{pmatrix} 0 & 0 & 0 & 0 & xz & -z^2 \\ 0 & 0 & 0 & 0 & xy & -yz \\ 0 & -x^2z + xyz + xz^2 & y^2z & -xz + yz & x - y & 0 \\ 0 & 0 & 0 & 0 & x^2 & -xz \\ -xy & -x^3 + x^2y + x^2z & xy^2 & -x^2 + xy & 0 & x - y \\ z & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \rangle}$$

Examples

$$M \cong \frac{R^{1 \times n}}{\langle r_1, \dots, r_m \rangle}$$

$$R = \mathbb{Q}[x, y, z]$$

$$\frac{\mathbb{Q}[x, y, z]}{\langle x^2 + y^2 + z^2 - 1 \rangle}$$

$$\frac{\mathbb{Q}[x, y, z]^{1 \times 2}}{\langle (x \ -y), (y \ -x) \rangle}$$

$$\frac{\mathbb{Q}[x, y, z]^{1 \times 6}}{\left\langle \begin{pmatrix} 0 & 0 & 0 & 0 & xz & -z^2 \\ 0 & 0 & 0 & 0 & xy & -yz \\ 0 & -x^2z + xyz + xz^2 & y^2z & -xz + yz & x - y & 0 \\ 0 & 0 & 0 & 0 & x^2 & -xz \\ -xy & -x^3 + x^2y + x^2z & xy^2 & -x^2 + xy & 0 & x - y \\ z & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \right\rangle}$$

Category of finitely presented modules

Finitely presented R -modules form a category

Category of finitely presented modules

Finitely presented R -modules form a category

$$\text{mod}_R$$

Category of finitely presented modules

Finitely presented R -modules form a category

$$\text{mod}_R$$

with R -linear maps as morphisms.

Category of finitely presented modules

Finitely presented R -modules form a category

$$\text{mod}_R$$

with R -linear maps as morphisms.

Computerfriendly model?

Applying the organizing principle

Goal: create computerfriendly model fpres_R of mod_R .

What we need

1 Data structures

- objects
- morphisms

2 Algorithms

- composition
- identities
- KernelObject
- ...

Applying the organizing principle

Goal: create computerfriendly model fpres_R of mod_R .

What we need

1 Data structures

- objects
- morphisms

2 Algorithms

- composition
- identities
- KernelObject
- ...

Applying the organizing principle

Goal: create computerfriendly model fpres_R of mod_R .

What we need

1 Data structures

- objects
- morphisms

2 Algorithms

- composition
- identities
- KernelObject
- ...

Data structures: objects

$$\langle \text{Rows of } \begin{array}{c} \mathbb{Q}[x, y, z]^{1 \times 6} \\ \hline \begin{pmatrix} 0 & 0 & 0 & 0 & xz & -z^2 \\ 0 & 0 & 0 & 0 & xy & -yz \\ 0 & -x^2z + xyz + xz^2 & y^2z & -xz + yz & x - y & 0 \\ 0 & 0 & 0 & 0 & x^2 & -xz \\ -xy & -x^3 + x^2y + x^2z & xy^2 & -x^2 + xy & 0 & x - y \\ z & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \rangle$$

Data structures: objects

$$\mathbb{Q}[x, y, z]^{1 \times 6}$$

$$\left\langle \begin{pmatrix} 0 & 0 & 0 & 0 & xz & -z^2 \\ 0 & 0 & 0 & 0 & xy & -yz \\ 0 & -x^2z + xyz + xz^2 & y^2z & -xz + yz & x - y & 0 \\ 0 & 0 & 0 & 0 & x^2 & -xz \\ -xy & -x^3 + x^2y + x^2z & xy^2 & -x^2 + xy & 0 & x - y \\ z & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \right\rangle$$

Data structures: objects

$$\mathbb{Q}[x, y, z]^{1 \times 6}$$

$$\left\langle \begin{pmatrix} 0 & 0 & 0 & 0 & xz & -z^2 \\ 0 & 0 & 0 & 0 & xy & -yz \\ 0 & -x^2z + xyz + xz^2 & y^2z & -xz + yz & x - y & 0 \\ 0 & 0 & 0 & 0 & x^2 & -xz \\ -xy & -x^3 + x^2y + x^2z & xy^2 & -x^2 + xy & 0 & x - y \\ z & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \right\rangle$$

Data structures: objects

$$\mathbb{Q}[x, y, z]^{1 \times 6}$$
$$\left\langle \begin{pmatrix} 0 & 0 & 0 & 0 & xz & -z^2 \\ 0 & 0 & 0 & 0 & xy & -yz \\ 0 & -x^2z + xyz + xz^2 & y^2z & -xz + yz & x - y & 0 \\ 0 & 0 & 0 & 0 & x^2 & -xz \\ -xy & -x^3 + x^2y + x^2z & xy^2 & -x^2 + xy & 0 & x - y \\ z & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \right\rangle$$

Idea: a matrix $M \in R^{m \times n}$ can represent the module $\frac{R^{1 \times n}}{\langle M \rangle}$.

Data structures: objects

$$\mathbb{Q}[x, y, z]^{1 \times 6}$$
$$\left\langle \begin{pmatrix} 0 & 0 & 0 & 0 & xz & -z^2 \\ 0 & 0 & 0 & 0 & xy & -yz \\ 0 & -x^2z + xyz + xz^2 & y^2z & -xz + yz & x - y & 0 \\ 0 & 0 & 0 & 0 & x^2 & -xz \\ -xy & -x^3 + x^2y + x^2z & xy^2 & -x^2 + xy & 0 & x - y \\ z & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \right\rangle$$

Idea: a matrix $M \in R^{m \times n}$ can represent the module $\frac{R^{1 \times n}}{\langle M \rangle}$.

Objects

$$\text{Obj}_{\text{fpres}_R} := \bigsqcup_{m, n \in \mathbb{N}_0} R^{m \times n}$$

Applying the organizing principle

Goal: create computerfriendly model fpres_R of mod_R .

What we need

1 Data structures

- objects
- morphisms

2 Algorithms

- composition
- identities
- KernelObject
- ...

Applying the organizing principle

Goal: create computerfriendly model fpres_R of mod_R .

What we need

1 Data structures

- objects
- morphisms

2 Algorithms

- composition
- identities
- KernelObject
- ...

Applying the organizing principle

Goal: create computerfriendly model fpres_R of mod_R .

What we need

1 Data structures

- objects ✓
- morphisms

2 Algorithms

- composition
- identities
- KernelObject
- ...

Applying the organizing principle

Goal: create computerfriendly model fpres_R of mod_R .

What we need

1 Data structures

- objects ✓
- morphisms

2 Algorithms

- composition
- identities
- KernelObject
- ...

Data structures: morphisms

Given: $M \in R^{m \times n}$ and $M' \in R^{m' \times n'}$.

Data structures: morphisms

Given: $M \in R^{m \times n}$ and $M' \in R^{m' \times n'}$.

$$\frac{R^{1 \times n}}{\langle M \rangle}$$

$$\frac{R^{1 \times n'}}{\langle M' \rangle}$$

Data structures: morphisms

Given: $M \in R^{m \times n}$ and $M' \in R^{m' \times n'}$.

$$\frac{R^{1 \times n}}{\langle M \rangle} \longrightarrow \frac{R^{1 \times n'}}{\langle M' \rangle}$$

Data structures: morphisms

Given: $M \in R^{m \times n}$ and $M' \in R^{m' \times n'}$.

$$\frac{R^{1 \times n}}{\langle M \rangle} \longrightarrow \frac{R^{1 \times n'}}{\langle M' \rangle}$$
$$\overline{e_j}$$

Data structures: morphisms

Given: $M \in R^{m \times n}$ and $M' \in R^{m' \times n'}$.

$$\begin{array}{ccc} \frac{R^{1 \times n}}{\langle M \rangle} & \longrightarrow & \frac{R^{1 \times n'}}{\langle M' \rangle} \\ \overline{e_i} & \longmapsto & \overline{r_i} \end{array}$$

Data structures: morphisms

Given: $M \in R^{m \times n}$ and $M' \in R^{m' \times n'}$.

$$\begin{array}{ccc} \frac{R^{1 \times n}}{\langle M \rangle} & \xrightarrow{\begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix}} & \frac{R^{1 \times n'}}{\langle M' \rangle} \\ \overline{e_i} & \longmapsto & \overline{r_i} \end{array}$$

Data structures: morphisms

Given: $M \in R^{m \times n}$ and $M' \in R^{m' \times n'}$.

$$\begin{array}{ccc} \frac{R^{1 \times n}}{\langle M \rangle} & \xrightarrow{\begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix}} & \frac{R^{1 \times n'}}{\langle M' \rangle} \\ \overline{e}_i & \longmapsto & \overline{r}_i \end{array}$$

$\text{Hom}_{\text{fpres}_R}(M, M') :=$

Data structures: morphisms

Given: $M \in R^{m \times n}$ and $M' \in R^{m' \times n'}$.

$$\begin{array}{ccc} & A := \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix} & \\ \frac{R^{1 \times n}}{\langle M \rangle} & \xrightarrow{\quad} & \frac{R^{1 \times n'}}{\langle M' \rangle} \\ \overline{e_i} & \longmapsto & \overline{r_i} \end{array}$$

$\text{Hom}_{\text{fpres}_R}(M, M') :=$

$$A \in R^{n \times n'}$$

Data structures: morphisms

Given: $M \in R^{m \times n}$ and $M' \in R^{m' \times n'}$.

$$\begin{array}{ccc} & A := \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix} & \\ \frac{R^{1 \times n}}{\langle M \rangle} & \xrightarrow{\quad} & \frac{R^{1 \times n'}}{\langle M' \rangle} \\ \overline{e_i} & \longmapsto & \overline{r_i} \end{array}$$

$\text{Hom}_{\text{fpres}_R}(M, M') :=$

$A \in R^{n \times n'}$ such that

Data structures: morphisms

Given: $M \in R^{m \times n}$ and $M' \in R^{m' \times n'}$.

$$\begin{array}{ccc} & A := \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix} & \\ \frac{R^{1 \times n}}{\langle M \rangle} & \xrightarrow{\quad} & \frac{R^{1 \times n'}}{\langle M' \rangle} \\ \overline{e}_i & \longmapsto & \overline{r}_i \end{array}$$

$\text{Hom}_{\text{fpres}_R}(M, M') :=$

$A \in R^{n \times n'}$ such that $\{\text{Rows of } M \cdot A\} \subseteq \langle M' \rangle$

Data structures: morphisms

Given: $M \in R^{m \times n}$ and $M' \in R^{m' \times n'}$.

$$\begin{array}{ccc} & A := \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix} & \\ \frac{R^{1 \times n}}{\langle M \rangle} & \xrightarrow{\quad} & \frac{R^{1 \times n'}}{\langle M' \rangle} \\ \overline{e}_i & \longmapsto & \overline{r}_i \end{array}$$

$\text{Hom}_{\text{fpres}_R}(M, M') :=$

$A \in R^{n \times n'}$ such that $\exists X \in R^{m \times m'} : M \cdot A = X \cdot M'$

Data structures: morphisms

Given: $M \in R^{m \times n}$ and $M' \in R^{m' \times n'}$.

$$\begin{array}{ccc} & A := \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix} & \\ \frac{R^{1 \times n}}{\langle M \rangle} & \xrightarrow{\quad} & \frac{R^{1 \times n'}}{\langle M' \rangle} \\ \overline{e}_i & \longmapsto & \overline{r}_i \end{array}$$

$\text{Hom}_{\text{fpres}_R}(M, M') :=$

$A \in R^{n \times n'}$ such that $\exists X \in R^{m \times m'} : M \cdot A = X \cdot M'$

A defines the 0 morphism

Data structures: morphisms

Given: $M \in R^{m \times n}$ and $M' \in R^{m' \times n'}$.

$$\begin{array}{ccc} & A := \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix} & \\ \frac{R^{1 \times n}}{\langle M \rangle} & \xrightarrow{\quad} & \frac{R^{1 \times n'}}{\langle M' \rangle} \\ \overline{e}_i & \longmapsto & \overline{r}_i \end{array}$$

$\text{Hom}_{\text{fpres}_R}(M, M') :=$

$A \in R^{n \times n'}$ such that $\exists X \in R^{m \times m'} : M \cdot A = X \cdot M'$

A defines the 0 morphism iff $\exists X \in R^{n \times m'} : A = X \cdot M'$

Data structures: morphisms

Given: $M \in R^{m \times n}$ and $M' \in R^{m' \times n'}$.

$$\begin{array}{ccc} & A := \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix} & \\ \frac{R^{1 \times n}}{\langle M \rangle} & \xrightarrow{\quad} & \frac{R^{1 \times n'}}{\langle M' \rangle} \\ \overline{e_i} & \longmapsto & \overline{r_i} \end{array}$$

$\text{Hom}_{\text{fpres}_R}(M, M') :=$

$A \in R^{n \times n'}$ such that $\exists X \in R^{m \times m'} : M \cdot A = X \cdot M'$

A defines the 0 morphism iff $\exists X \in R^{n \times m'} : A = X \cdot M'$

Data structures: morphisms

Deciding **well-definedness** and **being zero** of morphisms in fpres_R requires finding particular solutions of inhomogeneous linear systems.

Deciding **well-definedness** and **being zero** of morphisms in fpres_R requires finding particular solutions of inhomogeneous linear systems.

Ring	Algorithms
------	------------

Data structures: morphisms

Deciding **well-definedness** and **being zero** of morphisms in fpres_R requires finding particular solutions of inhomogeneous linear systems.

Ring	Algorithms
\mathbb{Q}	Gauss

Data structures: morphisms

Deciding **well-definedness** and **being zero** of morphisms in fpres_R requires finding particular solutions of inhomogeneous linear systems.

Ring	Algorithms
\mathbb{Q}	Gauss
\mathbb{Z}	Hermite

Data structures: morphisms

Deciding **well-definedness** and **being zero** of morphisms in fpres_R requires finding particular solutions of inhomogeneous linear systems.

Ring	Algorithms
\mathbb{Q}	Gauss
\mathbb{Z}	Hermite
$\mathbb{Q}[x, y, z]$	Buchberger

Applying the organizing principle

Goal: create computerfriendly model fpres_R of mod_R .

What we need

1 Data structures

- objects ✓
- morphisms

2 Algorithms

- composition
- identities
- KernelEmbedding
- ...

Applying the organizing principle

Goal: create computerfriendly model fpres_R of mod_R .

What we need

1 Data structures

- objects ✓
- morphisms

2 Algorithms

- composition
- identities
- KernelEmbedding
- ...

Applying the organizing principle

Goal: create computerfriendly model fpres_R of mod_R .

What we need

1 Data structures

- objects ✓
- morphisms ✓

2 Algorithms

- composition
- identities
- KernelEmbedding
- ...

Applying the organizing principle

Goal: create computerfriendly model fpres_R of mod_R .

What we need

1 Data structures ✓

- objects ✓
- morphisms ✓

2 Algorithms

- composition
- identities
- KernelEmbedding
- ...

Applying the organizing principle

Goal: create computerfriendly model fpres_R of mod_R .

What we need

1 Data structures ✓

- objects ✓
- morphisms ✓

2 Algorithms

- composition
- identities
- KernelEmbedding
- ...

Applying the organizing principle

Goal: create computerfriendly model fpres_R of mod_R .

What we need

1 Data structures ✓

- objects ✓
- morphisms ✓

2 Algorithms

- composition
- identities
- KernelEmbedding
- ...

Applying the organizing principle

Goal: create computerfriendly model fpres_R of mod_R .

What we need

1 Data structures ✓

- objects ✓
- morphisms ✓

2 Algorithms

- composition
- identities
- KernelEmbedding
- ...

Applying the organizing principle

Goal: create computerfriendly model fpres_R of mod_R .

What we need

1 Data structures ✓

- objects ✓
- morphisms ✓

2 Algorithms

- composition ✓
- identities
- KernelEmbedding
- ...

Applying the organizing principle

Goal: create computerfriendly model fpres_R of mod_R .

What we need

1 Data structures ✓

- objects ✓
- morphisms ✓

2 Algorithms

- composition ✓
- identities
- KernelEmbedding
- ...

Applying the organizing principle

Goal: create computerfriendly model fpres_R of mod_R .

What we need

1 Data structures ✓

- objects ✓
- morphisms ✓

2 Algorithms

- composition ✓
- identities
- KernelEmbedding
- ...

Applying the organizing principle

Goal: create computerfriendly model fpres_R of mod_R .

What we need

1 Data structures ✓

- objects ✓
- morphisms ✓

2 Algorithms

- composition ✓
- identities ✓
- KernelEmbedding
- ...

Applying the organizing principle

Goal: create computerfriendly model fpres_R of mod_R .

What we need

1 Data structures ✓

- objects ✓
- morphisms ✓

2 Algorithms

- composition ✓
- identities ✓
- KernelEmbedding
- ...

Applying the organizing principle

Goal: create computerfriendly model fpres_R of mod_R .

What we need

1 Data structures ✓

- objects ✓
- morphisms ✓

2 Algorithms

- composition ✓
- identities ✓
- [KernelEmbedding](#)
- ...

We cannot expect kernels to exist in fpres_R for all rings:

We cannot expect kernels to exist in fpres_R for all rings:

Example

$$R := k[Z, X_i \mid i \in \mathbb{N}] / \langle ZX_i \mid i \in \mathbb{N} \rangle.$$

We cannot expect kernels to exist in fpres_R for all rings:

Example

$R := k[Z, X_i \mid i \in \mathbb{N}] / \langle ZX_i \mid i \in \mathbb{N} \rangle$. Then

$$\alpha : R \longrightarrow R : r \mapsto \bar{Z} \cdot r$$

We cannot expect kernels to exist in fpres_R for all rings:

Example

$R := k[Z, X_i \mid i \in \mathbb{N}] / \langle ZX_i \mid i \in \mathbb{N} \rangle$. Then

$$\alpha : R \longrightarrow R : r \mapsto \bar{Z} \cdot r$$

has

$$\ker(\alpha) = \langle \bar{X}_i \mid i \in \mathbb{N} \rangle.$$

Kernels

Rings for which fpres_R has kernels are called **coherent**.

Rings for which fpres_R has kernels are called **coherent**.

Examples

Q

Kernels

Rings for which fpres_R has kernels are called **coherent**.

Examples

$\mathbb{Q}, \quad \mathbb{Z}$

Rings for which fpres_R has kernels are called **coherent**.

Examples

\mathbb{Q} , \mathbb{Z} , $\mathbb{Q}[x, y, z]$

Rings for which fpres_R has kernels are called **coherent**.

Examples

\mathbb{Q} , \mathbb{Z} , $\mathbb{Q}[x, y, z]$, $\mathbb{Q}[X_i \mid i \in \mathbb{N}]$

Theorem

Theorem

Let R be a ring equipped with the following algorithm:

Theorem

Let R be a ring equipped with the following algorithm:

- **Input:** $A \in R^{m \times n}$.

Theorem

Let R be a ring equipped with the following algorithm:

- **Input:** $A \in R^{m \times n}$.
- **Output:** $S \in R^{s \times m}$ s.t. the rows of S generate the row kernel of A .

Theorem

Let R be a ring equipped with the following algorithm:

- **Input:** $A \in R^{m \times n}$.
- **Output:** $S \in R^{s \times m}$ s.t. the rows of S generate the row kernel of A .

Then we have an algorithm for computing

Theorem

Let R be a ring equipped with the following algorithm:

- **Input:** $A \in R^{m \times n}$.
- **Output:** $S \in R^{s \times m}$ s.t. the rows of S generate the row kernel of A .

Then we have an algorithm for computing

- `KernelObject`

Theorem

Let R be a ring equipped with the following algorithm:

- **Input:** $A \in R^{m \times n}$.
- **Output:** $S \in R^{s \times m}$ s.t. the rows of S generate the row kernel of A .

Then we have an algorithm for computing

- KernelObject
- KernelEmbedding

Theorem

Let R be a ring equipped with the following algorithm:

- **Input:** $A \in R^{m \times n}$.
- **Output:** $S \in R^{s \times m}$ s.t. the rows of S generate the row kernel of A .

Then we have an algorithm for computing

- `KernelObject`
- `KernelEmbedding`

in fpres_R .

Theorem

Let R be a ring equipped with the following algorithm:

- **Input:** $A \in R^{m \times n}$.
- **Output:** $S \in R^{s \times m}$ s.t. the rows of S generate the row kernel of A .

Then we have an algorithm for computing

- `KernelObject`
- `KernelEmbedding`

in fpres_R .

Can you prove this theorem?

Categorical abstraction is a powerful organizing principle and computational tool.

- 1 What is categorical abstraction?
- 2 How can it be used as an organizing principle?
- 3 Why is it a computational tool?

Computing the intersection

Let $M_1 \subseteq N$ and $M_2 \subseteq N$ subobjects in an abelian category.

Computing the intersection

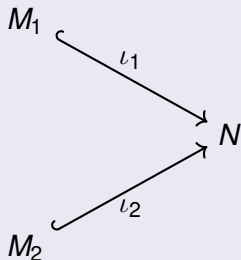
Let $M_1 \hookrightarrow N$ and $M_2 \hookrightarrow N$ subobjects in an abelian category.

Computing the intersection

Let $M_1 \hookrightarrow N$ and $M_2 \hookrightarrow N$ subobjects in an abelian category.
Compute their intersection $\gamma : M_1 \cap M_2 \hookrightarrow N$.

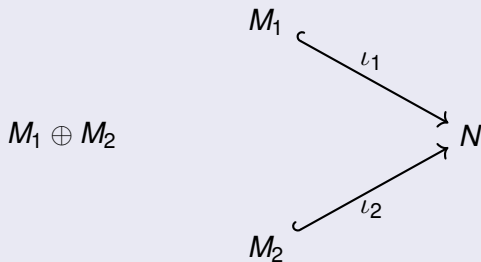
Computing the intersection

Let $M_1 \hookrightarrow N$ and $M_2 \hookrightarrow N$ subobjects in an abelian category.
Compute their intersection $\gamma : M_1 \cap M_2 \hookrightarrow N$.



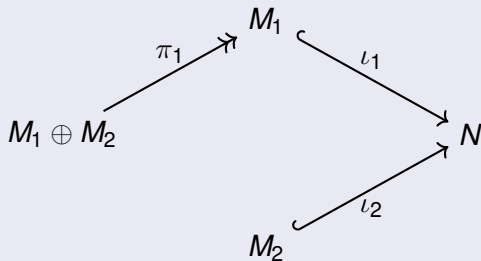
Computing the intersection

Let $M_1 \hookrightarrow N$ and $M_2 \hookrightarrow N$ subobjects in an abelian category.
Compute their intersection $\gamma : M_1 \cap M_2 \hookrightarrow N$.



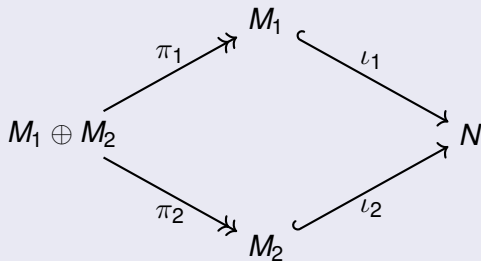
Computing the intersection

Let $M_1 \hookrightarrow N$ and $M_2 \hookrightarrow N$ subobjects in an abelian category.
Compute their intersection $\gamma : M_1 \cap M_2 \hookrightarrow N$.



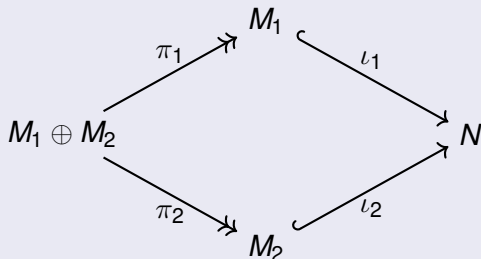
Computing the intersection

Let $M_1 \hookrightarrow N$ and $M_2 \hookrightarrow N$ subobjects in an abelian category.
Compute their intersection $\gamma : M_1 \cap M_2 \hookrightarrow N$.



Computing the intersection

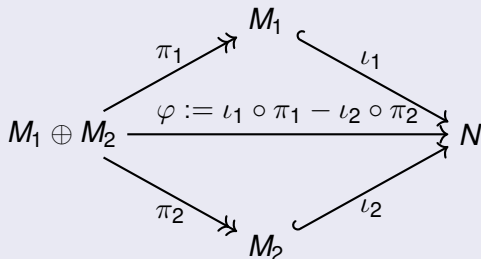
Let $M_1 \hookrightarrow N$ and $M_2 \hookrightarrow N$ subobjects in an abelian category.
Compute their intersection $\gamma : M_1 \cap M_2 \hookrightarrow N$.



- $\pi_i := \text{ProjectionInFactorOfDirectSum}((M_1, M_2), i), i = 1, 2$

Computing the intersection

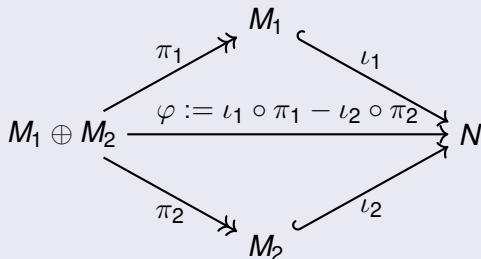
Let $M_1 \hookrightarrow N$ and $M_2 \hookrightarrow N$ subobjects in an abelian category.
Compute their intersection $\gamma : M_1 \cap M_2 \hookrightarrow N$.



- $\pi_i := \text{ProjectionInFactorOfDirectSum}((M_1, M_2), i), i = 1, 2$

Computing the intersection

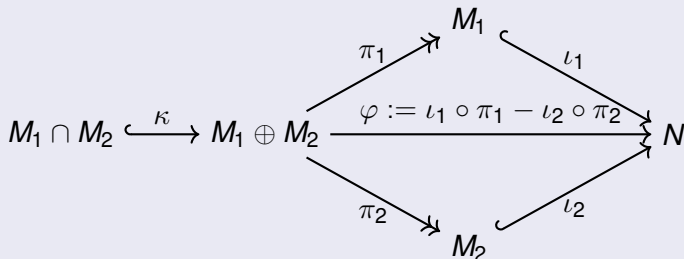
Let $M_1 \hookrightarrow N$ and $M_2 \hookrightarrow N$ subobjects in an abelian category.
Compute their intersection $\gamma : M_1 \cap M_2 \hookrightarrow N$.



- $\pi_i := \text{ProjectionInFactorOfDirectSum}((M_1, M_2), i), i = 1, 2$
- $\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$

Computing the intersection

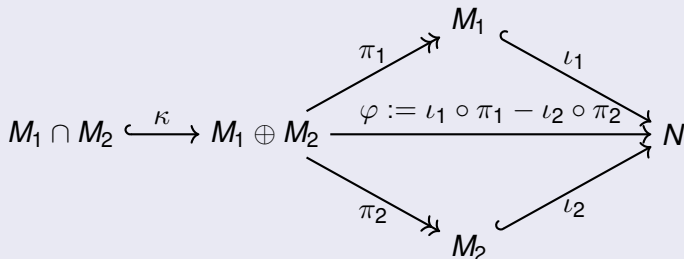
Let $M_1 \hookrightarrow N$ and $M_2 \hookrightarrow N$ subobjects in an abelian category.
Compute their intersection $\gamma : M_1 \cap M_2 \hookrightarrow N$.



- $\pi_i := \text{ProjectionInFactorOfDirectSum}((M_1, M_2), i), i = 1, 2$
- $\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$

Computing the intersection

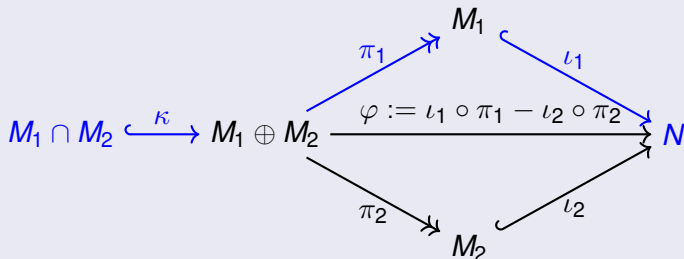
Let $M_1 \hookrightarrow N$ and $M_2 \hookrightarrow N$ subobjects in an abelian category.
Compute their intersection $\gamma : M_1 \cap M_2 \hookrightarrow N$.



- $\pi_i := \text{ProjectionInFactorOfDirectSum}((M_1, M_2), i), i = 1, 2$
- $\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$
- $\kappa := \text{KernelEmbedding}(\varphi)$

Computing the intersection

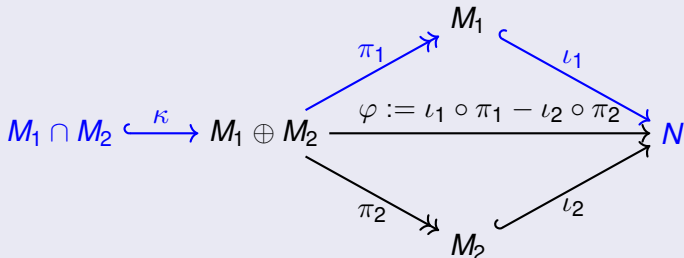
Let $M_1 \hookrightarrow N$ and $M_2 \hookrightarrow N$ subobjects in an abelian category.
Compute their intersection $\gamma : M_1 \cap M_2 \hookrightarrow N$.



- $\pi_i := \text{ProjectionInFactorOfDirectSum}((M_1, M_2), i), i = 1, 2$
- $\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$
- $\kappa := \text{KernelEmbedding}(\varphi)$

Computing the intersection

Let $M_1 \hookrightarrow N$ and $M_2 \hookrightarrow N$ subobjects in an abelian category.
Compute their intersection $\gamma : M_1 \cap M_2 \hookrightarrow N$.



- $\pi_i := \text{ProjectionInFactorOfDirectSum}((M_1, M_2), i), i = 1, 2$
- $\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$
- $\kappa := \text{KernelEmbedding}(\varphi)$
- $\gamma := \iota_1 \circ \pi_1 \circ \kappa$

Translation to CAP

$$\pi_i := \text{ProjectionInFactorOfDirectSum}((M_1, M_2), i), i = 1, 2$$

$$\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$$

$$\kappa := \text{KernelEmbedding}(\varphi)$$

$$\gamma := \iota_1 \circ \pi_1 \circ \kappa$$

Translation to CAP

$\pi_i := \text{ProjectionInFactorOfDirectSum}((M_1, M_2), i), i = 1, 2$

`pi1 := ProjectionInFactorOfDirectSum([M1, M2], 1);`

`pi2 := ProjectionInFactorOfDirectSum([M1, M2], 2);`

$\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$

$\kappa := \text{KernelEmbedding}(\varphi)$

$\gamma := \iota_1 \circ \pi_1 \circ \kappa$

Translation to CAP

$\pi_i := \text{ProjectionInFactorOfDirectSum}((M_1, M_2), i), i = 1, 2$

`pi1 := ProjectionInFactorOfDirectSum([M1, M2], 1);`

`pi2 := ProjectionInFactorOfDirectSum([M1, M2], 2);`

$\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$

`lambda := PostCompose(iota1, pi1);`

`phi := lambda - PostCompose(iota2, pi2);`

$\kappa := \text{KernelEmbedding}(\varphi)$

$\gamma := \iota_1 \circ \pi_1 \circ \kappa$

Translation to CAP

$\pi_i := \text{ProjectionInFactorOfDirectSum}((M_1, M_2), i), i = 1, 2$

`pi1 := ProjectionInFactorOfDirectSum([M1, M2], 1);`

`pi2 := ProjectionInFactorOfDirectSum([M1, M2], 2);`

$\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$

`lambda := PostCompose(iota1, pi1);`

`phi := lambda - PostCompose(iota2, pi2);`

$\kappa := \text{KernelEmbedding}(\varphi)$

`kappa := KernelEmbedding(phi);`

$\gamma := \iota_1 \circ \pi_1 \circ \kappa$

Translation to CAP

$\pi_i := \text{ProjectionInFactorOfDirectSum}((M_1, M_2), i), i = 1, 2$

`pi1 := ProjectionInFactorOfDirectSum([M1, M2], 1);`

`pi2 := ProjectionInFactorOfDirectSum([M1, M2], 2);`

$\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$

`lambda := PostCompose(iota1, pi1);`

`phi := lambda - PostCompose(iota2, pi2);`

$\kappa := \text{KernelEmbedding}(\varphi)$

`kappa := KernelEmbedding(phi);`

$\gamma := \iota_1 \circ \pi_1 \circ \kappa$

`gamma := PostCompose(lambda, kappa);`

Translation to CAP

```
pi1 := ProjectionInFactorOfDirectSum( [ M1, M2 ], 1 );  
pi2 := ProjectionInFactorOfDirectSum( [ M1, M2 ], 2 );
```

```
lambda := PostCompose( iota1, pi1 );  
phi := lambda - PostCompose( iota2, pi2 );
```

```
kappa := KernelEmbedding( phi );
```

```
gamma := PostCompose( lambda, kappa );
```

Translation to CAP

```
pi1 := ProjectionInFactorOfDirectSum( [ M1, M2 ], 1 );  
pi2 := ProjectionInFactorOfDirectSum( [ M1, M2 ], 2 );  
  
lambda := PostCompose( iota1, pi1 );  
phi := lambda - PostCompose( iota2, pi2 );  
  
kappa := KernelEmbedding( phi );  
  
gamma := PostCompose( lambda, kappa );
```

Translation to CAP

```
IntersectionSubobjects := function( iota1, iota2 )
```

```
    pi1 := ProjectionInFactorOfDirectSum( [ M1, M2 ], 1 );  
    pi2 := ProjectionInFactorOfDirectSum( [ M1, M2 ], 2 );  
  
    lambda := PostCompose( iota1, pi1 );  
    phi := lambda - PostCompose( iota2, pi2 );  
  
    kappa := KernelEmbedding( phi );  
  
    gamma := PostCompose( lambda, kappa );
```

Translation to CAP

```
IntersectionSubobjects := function( iota1, iota2 )
```

```
  M1 := Source( iota1 );
```

```
  M2 := Source( iota2 );
```

```
  pi1 := ProjectionInFactorOfDirectSum( [ M1, M2 ], 1 );
```

```
  pi2 := ProjectionInFactorOfDirectSum( [ M1, M2 ], 2 );
```

```
  lambda := PostCompose( iota1, pi1 );
```

```
  phi := lambda - PostCompose( iota2, pi2 );
```

```
  kappa := KernelEmbedding( phi );
```

```
  gamma := PostCompose( lambda, kappa );
```

Translation to CAP

```
IntersectionSubobjects := function( iota1, iota2 )

M1 := Source( iota1 );
M2 := Source( iota2 );

pi1 := ProjectionInFactorOfDirectSum( [ M1, M2 ], 1 );
pi2 := ProjectionInFactorOfDirectSum( [ M1, M2 ], 2 );

lambda := PostCompose( iota1, pi1 );
phi := lambda - PostCompose( iota2, pi2 );

kappa := KernelEmbedding( phi );

gamma := PostCompose( lambda, kappa );

return gamma;
end;
```

Translation to CAP

```
IntersectionSubobjects := function( iota1, iota2 )
  local M1, M2, pi1, pi2, lambda, phi, kappa, gamma;

  M1 := Source( iota1 );
  M2 := Source( iota2 );

  pi1 := ProjectionInFactorOfDirectSum( [ M1, M2 ], 1 );
  pi2 := ProjectionInFactorOfDirectSum( [ M1, M2 ], 2 );

  lambda := PostCompose( iota1, pi1 );
  phi := lambda - PostCompose( iota2, pi2 );

  kappa := KernelEmbedding( phi );

  gamma := PostCompose( lambda, kappa );

  return gamma;
end;
```

Computing the intersection

Compute the intersection in $\text{mat}_{\mathbb{Q}}$ of

$$\begin{array}{ccccc} M_1 & \xhookrightarrow{\iota_1 := \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}} & N & \xleftarrow{\iota_2 := \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}} & M_2 \\ \parallel & & \parallel & & \parallel \\ 2 & & 3 & & 2 \end{array}$$

Computing the intersection

Compute the intersection in $\text{mat}_{\mathbb{Q}}$ of

$$\begin{array}{ccccc} M_1 & \xhookrightarrow{\iota_1 := \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}} & N & \xleftarrow{\iota_2 := \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}} & M_2 \\ \parallel & & \parallel & & \parallel \\ 2 & & 3 & & 2 \end{array}$$

```
gap> gamma := IntersectionOfSubobject( iota1, iota2 );  
<A morphism in the category of matrices over Q>
```

Computing the intersection

Compute the intersection in $\text{mat}_{\mathbb{Q}}$ of

$$\begin{array}{ccccc} M_1 & \xhookrightarrow{\iota_1 := \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}} & N & \xleftarrow{\iota_2 := \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}} & M_2 \\ \parallel & & \parallel & & \parallel \\ 2 & & 3 & & 2 \end{array}$$

```
gap> gamma := IntersectionOfSubobject( iota1, iota2 );  
<A morphism in the category of matrices over Q>
```

```
gap> Display( gamma );  
[ [ 1, 1, 0 ] ]
```

A morphism in the category of matrices over \mathbb{Q}

Computing the intersection

The same algorithm can be applied in fpres_R

Computing the intersection

The same algorithm can be applied in fpres_R (your turn).

Goals of the hands-on session

After the hands-on session, you will be able to

Goals of the hands-on session

After the hands-on session, you will be able to

- define abelian categories constructively,

Goals of the hands-on session

After the hands-on session, you will be able to

- define abelian categories constructively,
- compute with f.d. vector spaces and f.p. modules only using categorical operations,

Goals of the hands-on session

After the hands-on session, you will be able to

- define abelian categories constructively,
- compute with f.d. vector spaces and f.p. modules only using categorical operations,
- write short generic code that works on every instance of an abelian category.

<https://github.com/sebastianpos/cap-aachen2018>

Goals of the hands-on session

After the hands-on session, you will be able to

- define abelian categories constructively,
- compute with f.d. vector spaces and f.p. modules only using categorical operations,
- write short generic code that works on every instance of an abelian category,

Goals of the hands-on session

After the hands-on session, you will be able to

- define abelian categories constructively,
 - `DirectSum`, `ProjectionInFactorOfDirectSum`,
`InjectionOfCofactorOfDirectSum`,
 - `KernelObject`, `KernelEmbedding`, `KernelLift`
 - `CokernelObject`, `CokernelProjection`, `CokernelColift`,
 - `LiftAlongMonomorphism`, `ColiftAlongEpimorphism`
- compute with f.d. vector spaces and f.p. modules only using categorical operations,
- write short generic code that works on every instance of an abelian category,

Goals of the hands-on session

After the hands-on session, you will be able to

- define abelian categories constructively,
 - `DirectSum`, `ProjectionInFactorOfDirectSum`,
`InjectionOfCofactorOfDirectSum`,
 - `KernelObject`, `KernelEmbedding`, `KernelLift`
 - `CokernelObject`, `CokernelProjection`, `CokernelColift`,
 - `LiftAlongMonomorphism`, `ColiftAlongEpimorphism`
- compute with f.d. vector spaces and f.p. modules only using categorical operations,
 - model for vector spaces: category of matrices,
 - model for modules: category of (left) presentations,
- write short generic code that works on every instance of an abelian category,

Goals of the hands-on session

After the hands-on session, you will be able to

- define abelian categories constructively,
 - `DirectSum`, `ProjectionInFactorOfDirectSum`, `InjectionOfCofactorOfDirectSum`,
 - `KernelObject`, `KernelEmbedding`, `KernelLift`
 - `CokernelObject`, `CokernelProjection`, `CokernelColift`,
 - `LiftAlongMonomorphism`, `ColiftAlongEpimorphism`
- compute with f.d. vector spaces and f.p. modules only using categorical operations,
 - model for vector spaces: category of matrices,
 - model for modules: category of (left) presentations,
- write short generic code that works on every instance of an abelian category,
 - functoriality of kernels,
 - intersection of subobjects,
 - addition of subobjects.

Take-home message

Categorical abstraction is a powerful organizing principle and computational tool.