How to implement a category on the computer and why?

Mohamed Barakat

Universität Siegen

EACA 2016
Logroño, Spain
June 24, 2016

Joint work with
Markus Lange-Hegermann, Sebastian Gutsche, Sebastian Posur

The algebra of morphisms

We will see why **category theory** is helpful in the development of **constructive mathematics**.

We will see why **category theory** is helpful in the development of **constructive mathematics**.

- A category $\mathcal{A}$ consists of

The algebra of morphisms

We will see why **category theory** is helpful in the development of **constructive mathematics**.

- A category $\mathcal{A}$ consists of
  - objects $L, M, N, \ldots$ and

The algebra of morphisms

We will see why **category theory** is helpful in the development of **constructive mathematics**.

- A category $\mathcal{A}$ consists of
  - objects $L, M, N, \ldots$ and
  - sets of morphisms $\mathrm{Hom}_{\mathcal{A}}(M, N)$.

The algebra of morphisms

We will see why **category theory** is helpful in the development of **constructive mathematics**.

- A category $\mathcal{A}$ consists of
  - objects $L, M, N, \dots$ and
  - sets of morphisms $\mathrm{Hom}_{\mathcal{A}}(M, N)$.
- In fact, only the Hom sets and their compositions are relevant

$$\begin{aligned} \mathrm{Hom}_{\mathcal{A}}(L, M) \times \mathrm{Hom}_{\mathcal{A}}(M, N) &\to \mathrm{Hom}_{\mathcal{A}}(L, N) \\ (\varphi, \psi) &\mapsto \varphi\psi. \end{aligned}$$

- This means that the notion “category” suppresses the “inner nature” of the objects and emphasizes the “algebra” of morphisms.

Equivalence of categories

- This means that the notion “category” suppresses the “inner nature” of the objects and emphasizes the “algebra” of morphisms.
- The objects are only place-holders, exactly like the vertices of a graph.

Equivalence of categories

- This means that the notion “category” suppresses the “inner nature” of the objects and emphasizes the “algebra” of morphisms.
- The objects are only place-holders, exactly like the vertices of a graph.
- The notion “equivalence of categories” gives one even more freedom in the description of a (constructive) model of the category.

Linear algebra and matrix theory

Here is a prominent example of this point of view.

Here is a prominent example of this point of view.

Example

Let k be a field. Then

$$k\text{-vec} := \begin{cases} \text{Obj:} & \text{finite dim. } k\text{-vector spaces,} \\ \text{Mor:} & k\text{-linear maps.} \end{cases}$$

Here is a prominent example of this point of view.

Example

Let k be a field. Then

$$k\text{-vec} := \begin{cases} \text{Obj:} & \text{finite dim. } k\text{-vector spaces,} \\ \text{Mor:} & k\text{-linear maps.} \end{cases}$$

$$\simeq$$

$$k\text{-mat} := \begin{cases} \text{Obj:} & \mathbb{N} \ni g, g', \ldots, \\ & \end{cases}$$

Here is a prominent example of this point of view.

Example

Let k be a field. Then

$$k\text{-vec} := \begin{cases} \text{Obj:} & \text{finite dim. } k\text{-vector spaces,} \\ \text{Mor:} & k\text{-linear maps.} \end{cases}$$

$$\simeq$$

$$k\text{-mat} := \begin{cases} \text{Obj:} & \mathbb{N} \ni g, g', \ldots, \\ \text{Mor:} & \mathtt{A} \in k^{g \times g'},\ g, g' \in \mathbb{N}. \end{cases}$$

Linear algebra and matrix theory

Here is a prominent example of this point of view.

Example

Let k be a field. Then

$$k\text{-vec} := \begin{cases} \text{Obj:} & \text{finite dim. } k\text{-vector spaces,} \\ \text{Mor:} & k\text{-linear maps.} \end{cases}$$

$$\simeq$$

$$k\text{-mat} := \begin{cases} \text{Obj:} & \mathbb{N} \ni g, g', \ldots, \\ \text{Mor:} & \mathtt{A} \in k^{g \times g'},\ g, g' \in \mathbb{N}. \end{cases}$$

$\rightsquigarrow$ from the categorical point of view, linear algebra and matrix theory are equivalent.

An ABELian category is a category in which we can do a very general form of linear algebra.

An ABELian category is a category in which we can do a very general form of linear algebra.

Definition

A category $\mathcal{A}$ is called ABELian if

ABELian categories

An ABELian category is a category in which we can do a very general form of linear algebra.

Definition

A category $\mathcal{A}$ is called ABELian if

- finite biproducts exist,

An ABELian category is a category in which we can do a very general form of linear algebra.

Definition

A category $\mathcal{A}$ is called ABELian if

- finite biproducts exist,
- each morphism has an additive inverse,

ABELian categories

An ABELian category is a category in which we can do a very general form of linear algebra.

Definition

A category $\mathcal{A}$ is called ABELian if

- finite biproducts exist,
- each morphism has an additive inverse,
- kernels and cokernels exist,

ABELian categories

An ABELian category is a category in which we can do a very general form of linear algebra.

Definition

A category $\mathcal{A}$ is called ABELian if

- finite biproducts exist,
- each morphism has an additive inverse,
- kernels and cokernels exist,
- the homomorphism theorem is valid, i.e., $\operatorname{coim} \varphi \xrightarrow{\sim} \operatorname{im} \varphi$.

ABELian categories

An ABELian category is a category in which we can do a very general form of linear algebra.

Definition

A category $\mathcal{A}$ is called ABELian if

- finite biproducts exist,
- each morphism has an additive inverse,
- kernels and cokernels exist,
- the homomorphism theorem is valid, i.e., $\operatorname{coim} \varphi \xrightarrow{\sim} \operatorname{im} \varphi$.

Definition

A category is called **constructively** ABELian if all disjunctions ($\vee$) and all existential quantifiers ($\exists$) in the axioms of an ABELian category are realized by algorithms.

The “hidden” existential quantifiers of “kernels”

Example

Let $\varphi : M \to N$ be a morphism in $\mathcal{A}$.

$$M \xrightarrow{\varphi} N$$

The “hidden” existential quantifiers of “kernels”

Example

Let $\varphi : M \to N$ be a morphism in $\mathcal{A}$.

$$\ker \varphi$$

$$M \xrightarrow{\varphi} N$$

The “hidden” existential quantifiers of “kernels”

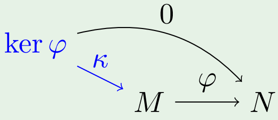
Example

Let $\varphi : M \to N$ be a morphism in $\mathcal{A}$.

$$\ker\varphi \xrightarrow{\kappa} M \xrightarrow{\varphi} N$$

Example

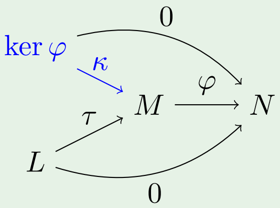
Let $\varphi : M \to N$ be a morphism in $\mathcal{A}$.



The “hidden” existential quantifiers of “kernels”

Example

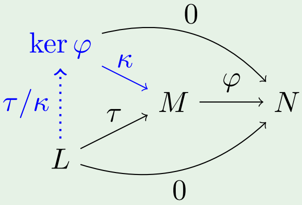
Let $\varphi : M \to N$ be a morphism in $\mathcal{A}$.



The “hidden” existential quantifiers of “kernels”

Example

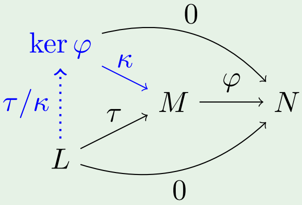
Let $\varphi : M \to N$ be a morphism in $\mathcal{A}$.



The "hidden" existential quantifiers of "kernels"

Example

Let $\varphi : M \to N$ be a morphism in $\mathcal{A}$.

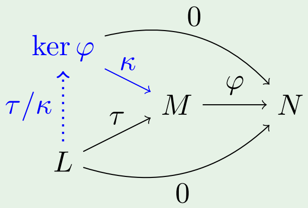


So $\mathcal{A}$ is a computational context with *many* basic algorithms.

The “hidden” existential quantifiers of “kernels”

Example

Let $\varphi : M \to N$ be a morphism in $\mathcal{A}$.



So $\mathcal{A}$ is a computational context with *many* basic algorithms.

Q:

Are module categories constructive, like $k\text{-}\mathrm{vec}$?

A constructive model for R-$\mathbf{mod}$

From now on let R be a ring with 1.

A constructive model for $R\text{-}\mathbf{mod}$

From now on let R be a ring with 1.

Definition

Let $\mathtt{A} \in R^{r \times c}$ and $\mathtt{B} \in R^{r' \times c}$ be two stackable matrices.

A constructive model for R-$\mathbf{mod}$

From now on let R be a ring with 1.

Definition

Let $\mathtt{A} \in R^{r \times c}$ and $\mathtt{B} \in R^{r' \times c}$ be two stackable matrices. We say that $\mathtt{A}$ **row-dominates** $\mathtt{B}$ if there exists a matrix $\mathtt{X}$ satisfying $\mathtt{X}\mathtt{A} = \mathtt{B}$.

A constructive model for R-$\mathbf{mod}$

From now on let R be a ring with 1.

Definition

Let $\mathtt{A} \in R^{r \times c}$ and $\mathtt{B} \in R^{r' \times c}$ be two stackable matrices. We say that $\mathtt{A}$ **row-dominates** $\mathtt{B}$ if there exists a matrix $\mathtt{X}$ satisfying $\mathtt{XA} = \mathtt{B}$. We write $\mathtt{A} \geq \mathtt{B}$.

A constructive model for R-$\mathbf{mod}$

From now on let R be a ring with 1.

Definition

Let $\mathtt{A} \in R^{r \times c}$ and $\mathtt{B} \in R^{r' \times c}$ be two stackable matrices. We say that $\mathtt{A}$ **row-dominates** $\mathtt{B}$ if there exists a matrix $\mathtt{X}$ satisfying $\mathtt{XA} = \mathtt{B}$. We write $\mathtt{A} \geq \mathtt{B}$.

Example

R-$\mathbf{mod} \simeq$

A constructive model for R-$\mathbf{mod}$

From now on let R be a ring with 1.

Definition

Let $\mathtt{A} \in R^{r \times c}$ and $\mathtt{B} \in R^{r' \times c}$ be two stackable matrices. We say that $\mathtt{A}$ **row-dominates** $\mathtt{B}$ if there exists a matrix $\mathtt{X}$ satisfying $\mathtt{XA} = \mathtt{B}$. We write $\mathtt{A} \geq \mathtt{B}$.

Example

R-$\mathbf{mod} \simeq$

$$R\text{-}\mathbf{fpres} := \begin{cases} \text{Obj:} \quad \mathtt{M} \in R^{r \times g}, \mathtt{N} \in R^{r' \times g'}, \ldots, \; r, g, r', g' \in \mathbb{N}, \\ \\ \\ \end{cases}$$

A constructive model for R-$\mathbf{mod}$

From now on let R be a ring with 1.

Definition

Let $\mathtt{A} \in R^{r \times c}$ and $\mathtt{B} \in R^{r' \times c}$ be two stackable matrices. We say that $\mathtt{A}$ **row-dominates** $\mathtt{B}$ if there exists a matrix $\mathtt{X}$ satisfying $\mathtt{XA} = \mathtt{B}$. We write $\mathtt{A} \geq \mathtt{B}$.

Example

$$R\text{-}\mathbf{mod} \simeq$$

$$R\text{-}\mathbf{fpres} := \begin{cases} \text{Obj:} & \mathtt{M} \in R^{r \times g}, \mathtt{N} \in R^{r' \times g'}, \ldots, \ r, g, r', g' \in \mathbb{N}, \\ \text{Mor:} & [(\mathtt{M}, \mathtt{A}, \mathtt{N})] \text{ with } \mathtt{A} \in R^{g \times g'} \text{ lies in } \mathrm{Hom}(\mathtt{M}, \mathtt{N}), \\ & \text{if } \mathtt{N} \geq \mathtt{MA}, \end{cases}$$

A constructive model for R-$\mathbf{mod}$

From now on let R be a ring with 1.

Definition

Let $\mathtt{A} \in R^{r \times c}$ and $\mathtt{B} \in R^{r' \times c}$ be two stackable matrices. We say that $\mathtt{A}$ **row-dominates** $\mathtt{B}$ if there exists a matrix $\mathtt{X}$ satisfying $\mathtt{XA} = \mathtt{B}$. We write $\mathtt{A} \geq \mathtt{B}$.

Example

$$R\text{-}\mathbf{mod} \simeq$$

$$R\text{-}\mathbf{fpres} := \begin{cases} \text{Obj:} & \mathtt{M} \in R^{r \times g}, \mathtt{N} \in R^{r' \times g'}, \ldots, \; r, g, r', g' \in \mathbb{N}, \\ \text{Mor:} & [(\mathtt{M}, \mathtt{A}, \mathtt{N})] \text{ with } \mathtt{A} \in R^{g \times g'} \text{ lies in } \mathrm{Hom}(\mathtt{M}, \mathtt{N}), \\ & \text{if } \mathtt{N} \geq \mathtt{MA}, \end{cases}$$

and $(\mathtt{M}, \mathtt{A}, \mathtt{N}) \sim (\mathtt{M}', \mathtt{A}', \mathtt{N}') :\Longleftrightarrow \mathtt{M} = \mathtt{M}', \mathtt{N} = \mathtt{N}', \mathtt{N} \geq \mathtt{A} - \mathtt{A}'$.

Computable rings

Definition

We call a constructive ring **left computable** if the solvability of $\mathtt{X}\mathtt{A} = \mathtt{B}$ is algorithmically decidable.

Computable rings

Definition

We call a constructive ring **left computable** if the solvability of $\mathtt{XA} = \mathtt{B}$ is algorithmically decidable.

Theorem ([BLH11])

If R is left computable then the category $R\text{-}\mathbf{fpres} \simeq R\text{-}\mathbf{mod}$ is constructively ABEL*ian.*

Examples of computable rings

Example (computable rings)

ring	algorithm
a constructive field k	GAUSS
ring of rational integers $\mathbb{Z}$	HERMITE normal form
a univariate polynomial ring $k[x]$	HERMITE normal form
a polynomial ring[a] $R[x_1, \ldots, x_n]$	BUCHBERGER
many noncommutative rings	n.c. BUCHBERGER
$k[x_1, \ldots, x_n]_{\langle x_1, \ldots, x_n \rangle}$	~~MORA~~ BUCHBERGER
residue class rings[b]	
...	

[a] R any of the above rings
[b] modulo ideals which are f.g. as left resp. right ideals.

In this context any algorithm to compute a GRÖBNER basis is a substitute for the GAUSS resp. HERMITE normal form algorithm.

Q:

Let R be a computable ring and $\mathfrak{p} \in \operatorname{Spec} R$ (finitely generated)

Q:

Let R be a computable ring and $\mathfrak{p} \in \operatorname{Spec} R$ (finitely generated):

- Is $R_{\mathfrak{p}}$ computable?

Q:

Let R be a computable ring and $\mathfrak{p} \in \operatorname{Spec} R$ (finitely generated):

- Is $R_\mathfrak{p}$ computable?
- Is $R_\mathfrak{p}$-$\mathbf{mod}$ constructive?

Q:

Let R be a computable ring and $\mathfrak{p} \in \operatorname{Spec} R$ (finitely generated):

- Is $R_{\mathfrak{p}}$ computable?
- Is $R_{\mathfrak{p}}$-$\mathbf{mod}$ constructive?

Even if there is a MORA algorithm ensuring the computability of the local ring $R_{\mathfrak{p}}$

Q:

Let R be a computable ring and $\mathfrak{p} \in \operatorname{Spec} R$ (finitely generated):

- Is $R_{\mathfrak{p}}$ computable?
- Is $R_{\mathfrak{p}}$-$\mathbf{mod}$ constructive?

Even if there is a MORA algorithm ensuring the computability of the local ring $R_{\mathfrak{p}}$ you still do not want to use it to establish the constructivity of $R_{\mathfrak{p}}$-$\mathbf{mod}$

Q:

Let R be a computable ring and $\mathfrak{p} \in \operatorname{Spec} R$ (finitely generated):

- Is $R_\mathfrak{p}$ computable?
- Is $R_\mathfrak{p}$-$\mathbf{mod}$ constructive?

Even if there is a MORA algorithm ensuring the computability of the local ring $R_\mathfrak{p}$ you still do not want to use it to establish the constructivity of $R_\mathfrak{p}$-$\mathbf{mod}$: It will be incredibly slow!

Q:

Let R be a computable ring and $\mathfrak{p} \in \operatorname{Spec} R$ (finitely generated):

- Is $R_\mathfrak{p}$ computable?
- Is $R_\mathfrak{p}$-$\mathbf{mod}$ constructive?

Even if there is a MORA algorithm ensuring the computability of the local ring $R_\mathfrak{p}$ you still do not want to use it to establish the constructivity of $R_\mathfrak{p}$-$\mathbf{mod}$: It will be incredibly slow!

Theorem

$$R_\mathfrak{p}\text{-}\mathbf{mod} \simeq R\text{-}\mathbf{mod}/\{M \in R\text{-}\mathbf{mod} \mid M_\mathfrak{p} = 0\}.$$

Q:

Let R be a computable ring and $\mathfrak{p} \in \operatorname{Spec} R$ (finitely generated):

- Is $R_\mathfrak{p}$ computable?
- Is $R_\mathfrak{p}$-$\mathbf{mod}$ constructive?

Even if there is a MORA algorithm ensuring the computability of the local ring $R_\mathfrak{p}$ you still do not want to use it to establish the constructivity of $R_\mathfrak{p}$-$\mathbf{mod}$: It will be incredibly slow!

Theorem

$$R_\mathfrak{p}\text{-}\mathbf{mod} \simeq R\text{-}\mathbf{mod}/\{M \in R\text{-}\mathbf{mod} \mid M_\mathfrak{p} = 0\}.$$

Equivalently, regard all morphisms φ in R-$\mathbf{mod}$ with $(\ker \varphi)_\mathfrak{p} = 0 = (\operatorname{coker} \varphi)_\mathfrak{p}$ as isomorphisms.

Q:

What about the "modules" supported on $D(\mathfrak{p}) = \operatorname{Spec} R \setminus V(\mathfrak{p})$?

Q:

What about the "modules" supported on $D(\mathfrak{p}) = \operatorname{Spec} R \setminus V(\mathfrak{p})$?

- Is the structure sheaf $\mathcal{O}_{D(\mathfrak{p})}$ "computable"?

Q:

What about the "modules" supported on $D(\mathfrak{p}) = \operatorname{Spec} R \setminus V(\mathfrak{p})$?

- Is the structure sheaf $\mathcal{O}_{D(\mathfrak{p})}$ "computable"?
- Is $\mathfrak{Coh}\, \mathcal{O}_{D(\mathfrak{p})}$ constructive?

Q:

What about the "modules" supported on $D(\mathfrak{p}) = \operatorname{Spec} R \setminus V(\mathfrak{p})$?

- Is the structure sheaf $\mathcal{O}_{D(\mathfrak{p})}$ "computable"?
- Is $\mathfrak{Coh}\, \mathcal{O}_{D(\mathfrak{p})}$ constructive?

Theorem

$$\mathfrak{Coh}\, \mathcal{O}_{D(\mathfrak{p})} \simeq R\text{-}\mathbf{mod}/\{\operatorname{Supp} M \subseteq V(\mathfrak{p})\}.$$

Q:

What about the "modules" supported on $D(\mathfrak{p}) = \operatorname{Spec} R \setminus V(\mathfrak{p})$?

- Is the structure sheaf $\mathcal{O}_{D(\mathfrak{p})}$ "computable"?
- Is $\mathfrak{Coh}\, \mathcal{O}_{D(\mathfrak{p})}$ constructive?

Theorem

$$\mathfrak{Coh}\, \mathcal{O}_{D(\mathfrak{p})} \simeq R\text{-}\mathbf{mod} / \{\operatorname{Supp} M \subseteq V(\mathfrak{p})\}.$$

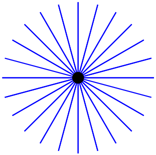
Equivalently, regard all morphisms φ in R-$\mathbf{mod}$ with $\operatorname{Supp}(\ker \varphi), \operatorname{Supp}(\operatorname{coker} \varphi) \subseteq V(\mathfrak{p})$ as isomorphisms.

Coherent sheaves on $\mathbb{P}^n$

The projective space $\mathbb{P}^n = \mathbb{P}V$ over a field k is the set of 1-dimensional subspaces of $V := k^{n+1}$

Coherent sheaves on $\mathbb{P}^n$

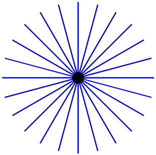
The projective space $\mathbb{P}^n = \mathbb{P}V$ over a field k is the set of 1-dimensional subspaces of $V := k^{n+1}$, i.e., the orbit space



$$V/k^* = \mathbb{P}^n \,\dot\cup\, \underbrace{\{0\}}_{\text{irrelevant locus}}$$

Coherent sheaves on $\mathbb{P}^n$

The projective space $\mathbb{P}^n = \mathbb{P}V$ over a field k is the set of 1-dimensional subspaces of $V := k^{n+1}$, i.e., the orbit space

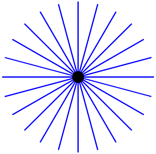


$$V/k^* = \mathbb{P}^n \,\dot\cup\, \underbrace{\{0\}}_{\text{irrelevant locus}}$$

A coherent sheaf (of modules) $\mathcal{F} = \widetilde{M}$ on $\mathbb{P}^n$ is informally

Coherent sheaves on $\mathbb{P}^n$

The projective space $\mathbb{P}^n = \mathbb{P}V$ over a field k is the set of 1-dimensional subspaces of $V := k^{n+1}$, i.e., the orbit space



$$V/k^* = \mathbb{P}^n \,\dot\cup\, \underbrace{\{0\}}_{\text{irrelevant locus}}$$

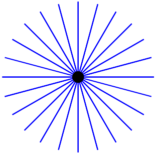
A coherent sheaf (of modules) $\mathcal{F} = \widetilde{M}$ on $\mathbb{P}^n$ is informally

- a f.g. module M over the polynomial ring

$$S := k[x_0, \ldots, x_n] = \operatorname{Sym} V^*$$

Coherent sheaves on $\mathbb{P}^n$

The projective space $\mathbb{P}^n = \mathbb{P}V$ over a field k is the set of 1-dimensional subspaces of $V := k^{n+1}$, i.e., the orbit space



$$V/k^* = \mathbb{P}^n \,\dot{\cup}\, \underbrace{\{0\}}_{\substack{\text{irrelevant}\\ \text{locus}}}$$

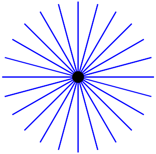
A coherent sheaf (of modules) $\mathcal{F} = \widetilde{M}$ on $\mathbb{P}^n$ is informally

- a f.g. module M over the polynomial ring

$$S := k[x_0, \ldots, x_n] = \operatorname{Sym} V^*$$

(viewed as a coherent sheaf on the affine space V)

The projective space $\mathbb{P}^n = \mathbb{P}V$ over a field k is the set of 1-dimensional subspaces of $V := k^{n+1}$, i.e., the orbit space



$$V/k^* = \mathbb{P}^n \,\dot\cup\, \underbrace{\{0\}}_{\text{irrelevant locus}}$$

A coherent sheaf (of modules) $\mathcal{F} = \widetilde{M}$ on $\mathbb{P}^n$ is informally

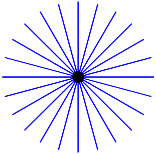
- a f.g. module M over the polynomial ring

$$S := k[x_0, \ldots, x_n] = \operatorname{Sym} V^*$$

  (viewed as a coherent sheaf on the affine space V),
- which is compatible with the action of k^*

Coherent sheaves on $\mathbb{P}^n$

The projective space $\mathbb{P}^n = \mathbb{P}V$ over a field k is the set of 1-dimensional subspaces of $V := k^{n+1}$, i.e., the orbit space



$$V/k^* = \mathbb{P}^n \dot\cup \underbrace{\{0\}}_{\text{irrelevant locus}}$$

A coherent sheaf (of modules) $\mathcal{F} = \widetilde{M}$ on $\mathbb{P}^n$ is informally

- a f.g. module M over the polynomial ring

$$S := k[x_0, \ldots, x_n] = \operatorname{Sym} V^*$$

  (viewed as a coherent sheaf on the affine space V),
- which is compatible with the action of k^*, and
- where S-modules supported on zero are treated as zero.

The compatibility with the k^* action $\rightsquigarrow M$ is a *graded* S-module $M = \bigoplus_{p \in \mathbb{Z}} M_p$ (S graded by $\deg$).

Coherent sheaves on $\mathbb{P}^n$

The compatibility with the k^* action $\rightsquigarrow$ M is a *graded* S-module $M = \bigoplus_{p\in\mathbb{Z}} M_p$ (S graded by $\deg$).

Theorem (Serre '55, FAC)

$$\mathfrak{Coh}\,\mathbb{P}^n = S\text{-grmod}/S\text{-grmod}^0,$$

Coherent sheaves on $\mathbb{P}^n$

The compatibility with the k^* action $\rightsquigarrow M$ is a *graded* S-module $M = \bigoplus_{p\in\mathbb{Z}} M_p$ (S graded by $\deg$).

Theorem (Serre '55, FAC)

$$\mathfrak{Coh}\,\mathbb{P}^n = S\text{-grmod}/S\text{-grmod}^0,$$

where S-grmod denotes the ABEL*ian category of f.g. graded S-modules*

Coherent sheaves on $\mathbb{P}^n$

The compatibility with the k^* action $\rightsquigarrow$ M is a *graded* S-module $M = \bigoplus_{p \in \mathbb{Z}} M_p$ (S graded by $\deg$).

Theorem (Serre '55, FAC)

$$\mathfrak{Coh}\,\mathbb{P}^n = S\text{-grmod}/S\text{-grmod}^0,$$

where S-grmod *denotes the* ABEL*ian category of f.g. graded* S*-modules and* S-grmod^0 *the subcategory of those supported on zero.*

SERRE quotient category

Definition

Let $\mathcal{A}$ be an ABELian category.

Definition

Let $\mathcal{A}$ be an ABELian category. A non-empty full subcategory $\mathcal{C} \subset \mathcal{A}$ is called **thick**

SERRE quotient category

Definition

Let $\mathcal{A}$ be an ABELian category. A non-empty full subcategory $\mathcal{C} \subset \mathcal{A}$ is called **thick** if it is closed under passing to subobjects, factor objects, and extensions.

SERRE quotient category

Definition

Let $\mathcal{A}$ be an ABELian category. A non-empty full subcategory $\mathcal{C} \subset \mathcal{A}$ is called **thick** if it is closed under passing to subobjects, factor objects, and extensions.

Example

$\mathcal{C} = S\text{-grmod}^0$ is a thick subcategory of $\mathcal{A} = S\text{-grmod}$.

SERRE quotient category

Definition

Let $\mathcal{A}$ be an ABELian category. A non-empty full subcategory $\mathcal{C} \subset \mathcal{A}$ is called **thick** if it is closed under passing to subobjects, factor objects, and extensions.

Example

$\mathcal{C} = S\text{-grmod}^0$ is a thick subcategory of $\mathcal{A} = S\text{-grmod}$.

Definition

The **SERRE quotient** $\mathcal{A}/\mathcal{C}$ is a category with

- $\operatorname{Obj} \mathcal{A}/\mathcal{C} := \operatorname{Obj} \mathcal{A}$;

Definition

Let $\mathcal{A}$ be an ABELian category. A non-empty full subcategory $\mathcal{C} \subset \mathcal{A}$ is called **thick** if it is closed under passing to subobjects, factor objects, and extensions.

Example

$\mathcal{C} = S\text{-grmod}^0$ is a thick subcategory of $\mathcal{A} = S\text{-grmod}$.

Definition

The **SERRE quotient** $\mathcal{A}/\mathcal{C}$ is a category with

- $\mathrm{Obj}\, \mathcal{A}/\mathcal{C} := \mathrm{Obj}\, \mathcal{A}$;
- $\mathrm{Hom}_{\mathcal{A}/\mathcal{C}}(M, N) := \varinjlim_{\substack{M' \leq M, N' \leq N, \\ M/M', N' \in \mathcal{C}}} \mathrm{Hom}_{\mathcal{A}}(M', N/N')$.

SERRE quotient category

Definition

Let $\mathcal{A}$ be an ABELian category. A non-empty full subcategory $\mathcal{C} \subset \mathcal{A}$ is called **thick** if it is closed under passing to subobjects, factor objects, and extensions.

Example

$\mathcal{C} = S\text{-grmod}^0$ is a thick subcategory of $\mathcal{A} = S\text{-grmod}$.

Definition

The **SERRE quotient** $\mathcal{A}/\mathcal{C}$ is a category with

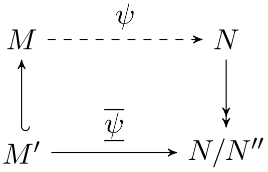
- $\mathrm{Obj}\,\mathcal{A}/\mathcal{C} := \mathrm{Obj}\,\mathcal{A}$;
- $\mathrm{Hom}_{\mathcal{A}/\mathcal{C}}(M,N) := \varinjlim_{\substack{M' \leq M, N' \leq N, \\ M/M', N' \in \mathcal{C}}} \mathrm{Hom}_{\mathcal{A}}(M', N/N')$.

$\mathcal{A}/\mathcal{C}$ is again ABELian and the **localization functor** $\mathscr{Q} : \mathcal{A} \to \mathcal{A}/\mathcal{C}, M \mapsto M, \varphi \mapsto [\varphi]$ is *exact*.

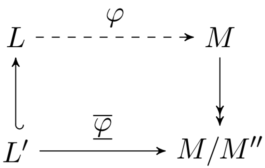
Constructive SERRE quotients

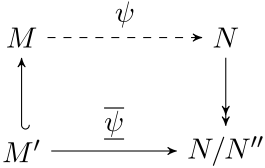
$$M \overset{\psi}{\dashrightarrow} N$$

Constructive SERRE quotients

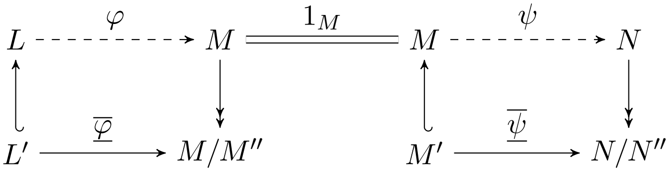


Constructive SERRE quotients





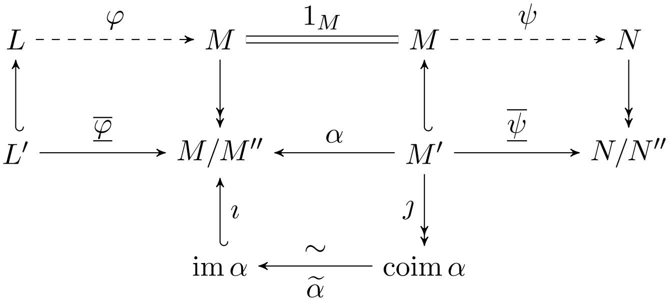
Constructive SERRE quotients

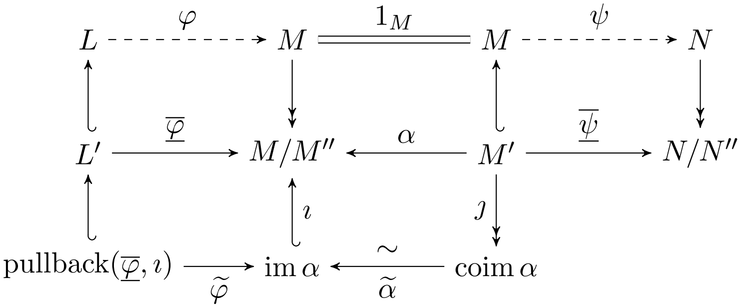


Constructive SERRE quotients

$$
\begin{array}{ccccccc}
L & \overset{\varphi}{\dashrightarrow} & M & \overset{1_M}{=\!=\!=} & M & \overset{\psi}{\dashrightarrow} & N \\
\uparrow & & \downarrow & & \uparrow & & \downarrow \\
L' & \xrightarrow{\overline{\varphi}} & M/M'' & \xleftarrow{\alpha} & M' & \xrightarrow{\overline{\psi}} & N/N''
\end{array}
$$

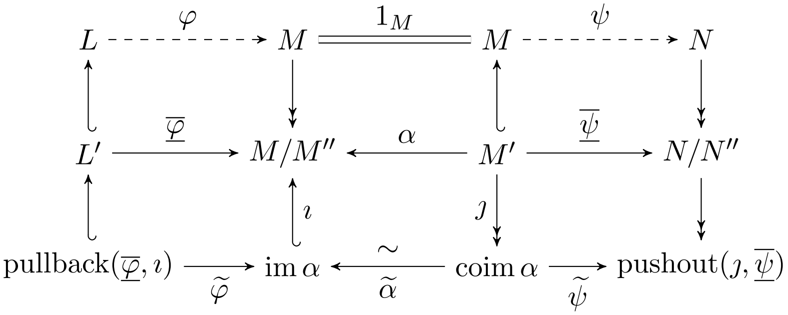
Constructive SERRE quotients



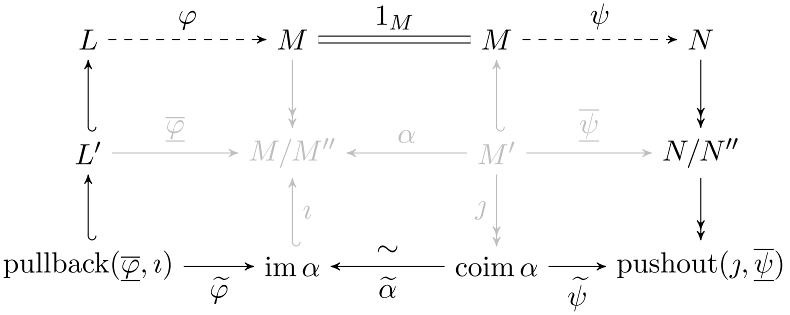


L
φ
M
1_M
M
ψ
N
L'
$\overline{\varphi}$
M/M''
α
M'
$\overline{\psi}$
N/N''
$\imath$
$\jmath$
pullback$(\overline{\varphi}, \imath)$
$\widetilde{\varphi}$
$\operatorname{im}\alpha$
$\sim$
$\widetilde{\alpha}$
$\operatorname{coim}\alpha$

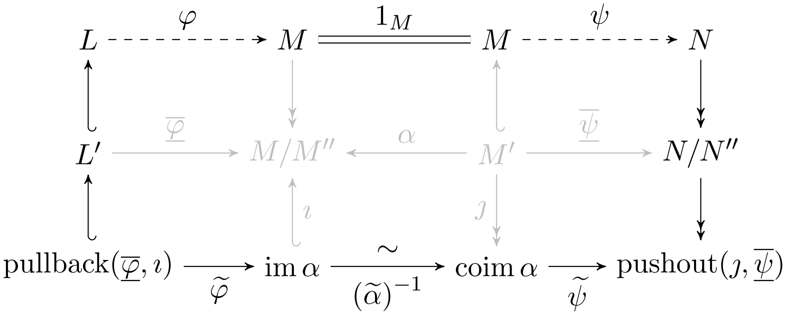
Constructive SERRE quotients



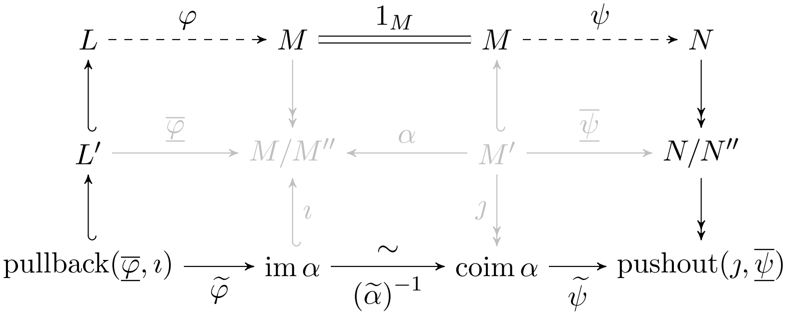
Constructive SERRE quotients



Constructive SERRE quotients



Constructive Serre quotients



Theorem ([BLH14])

Let $\mathcal{C} \subset \mathcal{A}$ be a thick subcategory of the Abel*ian category $\mathcal{A}$. If $\mathcal{A}$ is constructively* Abel*ian an the membership in $\mathcal{C}$ is decidable, then $\mathcal{A}/\mathcal{C}$ is constructively* Abel*ian.*

Corollaries

Corollary

$$R_{\mathfrak{p}}\text{-}\mathbf{mod} \simeq R\text{-}\mathbf{mod}/\{M \in R\text{-}\mathbf{mod} \mid M_{\mathfrak{p}} = 0\}$$

is constructively ABEL*ian.*

Corollaries

Corollary

$$R_{\mathfrak{p}}\text{-}\mathbf{mod} \simeq R\text{-}\mathbf{mod}/\{M \in R\text{-}\mathbf{mod} \mid M_{\mathfrak{p}} = 0\}$$

is constructively ABEL*ian.*

Corollary

$$\mathfrak{Coh}\, \mathcal{O}_{D(\mathfrak{p})} \simeq R\text{-}\mathbf{mod}/\{\operatorname{Supp} M \subseteq V(\mathfrak{p})\}$$

is constructively ABEL*ian.*

Corollaries

Corollary

$$R_{\mathfrak{p}}\text{-}\mathbf{mod} \simeq R\text{-}\mathbf{mod}/\{M \in R\text{-}\mathbf{mod} \mid M_{\mathfrak{p}} = 0\}$$

is constructively Abel*ian.*

Corollary

$$\mathfrak{Coh}\,\mathcal{O}_{D(\mathfrak{p})} \simeq R\text{-}\mathbf{mod}/\{\operatorname{Supp} M \subseteq V(\mathfrak{p})\}$$

is constructively Abel*ian.*

Corollary

$$\mathfrak{Coh}\,\mathbb{P}^n = S\text{-grmod}/S\text{-grmod}^0$$

is constructively Abel*ian.*

How to implement a category on the computer?

How to implement a category on the computer?

How to implement a category on the computer?

As a category is for the machine a computational context with *many* algorithms

How to implement a category on the computer?

How to implement a category on the computer?

As a category is for the machine a computational context with *many* algorithms we need an **object oriented** programing language

How to implement a category on the computer?

How to implement a category on the computer?

As a category is for the machine a computational context with *many* algorithms we need an **object oriented** programing language to

- implement the computational context as an object

How to implement a category on the computer?

As a category is for the machine a computational context with *many* algorithms we need an **object oriented** programing language to

- implement the computational context as an object and
- implement the *many* algorithms as associated methods.

How to implement a category on the computer?

How to implement a category on the computer?

As a category is for the machine a computational context with *many* algorithms we need an **object oriented** programing language to

- implement the computational context as an object and
- implement the *many* algorithms as associated methods.

As categorical constructions correspond to the passage from one computational context to another

How to implement a category on the computer?

As a category is for the machine a computational context with *many* algorithms we need an **object oriented** programing language to

- implement the computational context as an object and
- implement the *many* algorithms as associated methods.

As categorical constructions correspond to the passage from one computational context to another we also need a **functional** programing language

How to implement a category on the computer?

How to implement a category on the computer?

As a category is for the machine a computational context with *many* algorithms we need an **object oriented** programing language to

- implement the computational context as an object and
- implement the *many* algorithms as associated methods.

As categorical constructions correspond to the passage from one computational context to another we also need a **functional** programing language in order to

- take an algorithmic context as input and create another one out of it.

An example

Consider the category of finitely presented modules over the computable commutative ring $R := \mathbb{Q}[x, y, z]$:

An example

Consider the category of finitely presented modules over the computable commutative ring $R := \mathbb{Q}[x, y, z]$:

```
gap> Q := HomalgFieldOfRationalsInSingular( );
Q
```

An example

Consider the category of finitely presented modules over the computable commutative ring $R := \mathbb{Q}[x, y, z]$:

```
gap> Q := HomalgFieldOfRationalsInSingular( );
Q
gap> R := PolynomialRing( Q, "x,y,z" );
Q[x,y,z]
```

An example

Consider the category of finitely presented modules over the computable commutative ring $R := \mathbb{Q}[x, y, z]$:

```
gap> Q := HomalgFieldOfRationalsInSingular( );
Q
gap> R := PolynomialRing( Q, "x,y,z" );
Q[x,y,z]
gap> B := LeftPresentations( R );
The category of f.p. modules over Q[x,y,z]
```

An example

Consider the category of finitely presented modules over the computable commutative ring $R := \mathbb{Q}[x, y, z]$:

```
gap> Q := HomalgFieldOfRationalsInSingular( );
Q
gap> R := PolynomialRing( Q, "x,y,z" );
Q[x,y,z]
gap> B := LeftPresentations( R );
The category of f.p. modules over Q[x,y,z]
gap> InfoOfInstalledOperationsOfCategory( B );
34 primitive operations were used to derive 166 basic ones for \
this symmetric closed monoidal Abelian category
```

An example (continued)

But we can also construct new categories out of previously constructed ones:

An example (continued)

But we can also construct new categories out of previously constructed ones:

```
gap> S := GradedRing( R );;
```

An example (continued)

But we can also construct new categories out of previously constructed ones:

```
gap> S := GradedRing( R );;
gap> A := GradedLeftPresentations( S );
The category of f.p. graded modules over Q[x,y,z]
```

An example (continued)

But we can also construct new categories out of previously constructed ones:

```
gap> S := GradedRing( R );;
gap> A := GradedLeftPresentations( S );
The category of f.p. graded modules over Q[x,y,z]
gap> C := Subcategory( A, M -> HilbertPolynomial( M ) = 0 );;
```

An example (continued)

But we can also construct new categories out of previously constructed ones:

```
gap> S := GradedRing( R );;
gap> A := GradedLeftPresentations( S );
The category of f.p. graded modules over Q[x,y,z]
gap> C := Subcategory( A, M -> HilbertPolynomial( M ) = 0 );;
gap> CohP2 := A / C;
A Serre quotient of the category of f.p. graded modules \
over Q[x,y,z]
```

An example (continued)

But we can also construct new categories out of previously constructed ones:

```
gap> S := GradedRing( R );;
gap> A := GradedLeftPresentations( S );
The category of f.p. graded modules over Q[x,y,z]
gap> C := Subcategory( A, M -> HilbertPolynomial( M ) = 0 );;
gap> CohP2 := A / C;
A Serre quotient of the category of f.p. graded modules \
over Q[x,y,z]
gap> InfoOfInstalledOperationsOfCategory( CohP2 );
19 primitive operations were used to derive 129 basic ones for \
this Abelian category
```

An example (continued)

But we can also construct new categories out of previously constructed ones:

```
gap> S := GradedRing( R );;
gap> A := GradedLeftPresentations( S );
The category of f.p. graded modules over Q[x,y,z]
gap> C := Subcategory( A, M -> HilbertPolynomial( M ) = 0 );;
gap> CohP2 := A / C;
A Serre quotient of the category of f.p. graded modules \
over Q[x,y,z]
gap> InfoOfInstalledOperationsOfCategory( CohP2 );
19 primitive operations were used to derive 129 basic ones for \
this Abelian category
```

We have created a computational context (the category $\mathcal{A}$) and transformed it into another one (the category $\mathcal{A}/\mathcal{C} \simeq \mathfrak{Coh}\,\mathbb{P}^2$).

Quasi-isomorphisms

Definition

- A chain morphism $\mu_\bullet : (M_\bullet, \partial_\bullet^M) \to (N_\bullet, \partial_\bullet^N)$ is a **quasi-isomorphism** if it induces isomorphisms on homology: $\mu_i : H_i(M_\bullet) \xrightarrow{\sim} H_i(N_\bullet)$ for all i.

Quasi-isomorphisms

Definition

- A chain morphism $\mu_\bullet : (M_\bullet, \partial_\bullet^M) \to (N_\bullet, \partial_\bullet^N)$ is a **quasi-isomorphism** if it induces isomorphisms on homology: $\mu_i : H_i(M_\bullet) \xrightarrow{\sim} H_i(N_\bullet)$ for all i.
- Two complexes $(M_\bullet, \partial_\bullet^M), (N_\bullet, \partial_\bullet^N)$ are called **quasi-isomorphic** if there exists a quasi-isomorphism $\mu_\bullet : (M_\bullet, \partial_\bullet^M) \to (N_\bullet, \partial_\bullet^N)$.

Quasi-isomorphisms

Definition

- A chain morphism $\mu_\bullet : (M_\bullet, \partial_\bullet^M) \to (N_\bullet, \partial_\bullet^N)$ is a **quasi-isomorphism** if it induces isomorphisms on homology: $\mu_i : H_i(M_\bullet) \xrightarrow{\sim} H_i(N_\bullet)$ for all i.
- Two complexes $(M_\bullet, \partial_\bullet^M), (N_\bullet, \partial_\bullet^N)$ are called **quasi-isomorphic** if there exists a quasi-isomorphism $\mu_\bullet : (M_\bullet, \partial_\bullet^M) \to (N_\bullet, \partial_\bullet^N)$.

Being quasi-isomorphic is reflexiv and transitive

Quasi-isomorphisms

Definition

- A chain morphism $\mu_\bullet : (M_\bullet, \partial^M_\bullet) \to (N_\bullet, \partial^N_\bullet)$ is a **quasi-isomorphism** if it induces isomorphisms on homology: $\mu_i : H_i(M_\bullet) \xrightarrow{\sim} H_i(N_\bullet)$ for all i.
- Two complexes $(M_\bullet, \partial^M_\bullet), (N_\bullet, \partial^N_\bullet)$ are called **quasi-isomorphic** if there exists a quasi-isomorphism $\mu_\bullet : (M_\bullet, \partial^M_\bullet) \to (N_\bullet, \partial^N_\bullet)$.

Being quasi-isomorphic is reflexiv and transitive but not (yet) symmetric!

Quasi-isomorphisms

Definition

- A chain morphism $\mu_\bullet : (M_\bullet, \partial^M_\bullet) \to (N_\bullet, \partial^N_\bullet)$ is a **quasi-isomorphism** if it induces isomorphisms on homology: $\mu_i : H_i(M_\bullet) \xrightarrow{\sim} H_i(N_\bullet)$ for all i.
- Two complexes $(M_\bullet, \partial^M_\bullet), (N_\bullet, \partial^N_\bullet)$ are called **quasi-isomorphic** if there exists a quasi-isomorphism $\mu_\bullet : (M_\bullet, \partial^M_\bullet) \to (N_\bullet, \partial^N_\bullet)$.

Being quasi-isomorphic is reflexiv and transitive but not (yet) symmetric!

Example

Regarding $\mathcal{A} \subset \mathbf{C}(\mathcal{A})$

$$\cdots \longleftarrow 0 \longleftarrow M \longleftarrow 0 \longleftarrow \cdots$$

Quasi-isomorphisms

Definition

- A chain morphism $\mu_\bullet : (M_\bullet, \partial^M_\bullet) \to (N_\bullet, \partial^N_\bullet)$ is a **quasi-isomorphism** if it induces isomorphisms on homology: $\mu_i : H_i(M_\bullet) \xrightarrow{\sim} H_i(N_\bullet)$ for all i.
- Two complexes $(M_\bullet, \partial^M_\bullet), (N_\bullet, \partial^N_\bullet)$ are called **quasi-isomorphic** if there exists a quasi-isomorphism $\mu_\bullet : (M_\bullet, \partial^M_\bullet) \to (N_\bullet, \partial^N_\bullet)$.

Being quasi-isomorphic is reflexiv and transitive but not (yet) symmetric!

Example

Regarding $\mathcal{A} \subset \mathbf{C}(\mathcal{A})$ resolutions become quasi-isomorphisms:

$$\begin{array}{ccccccccc}
\cdots & \longleftarrow & 0 & \longleftarrow & P_0 & \longleftarrow & P_1 & \longleftarrow & \cdots \\
 & & \downarrow & & \downarrow \pi & & \downarrow & & \\
\cdots & \longleftarrow & 0 & \longleftarrow & M & \longleftarrow & 0 & \longleftarrow & \cdots
\end{array}$$

Homotopy equivalences

Definition

- Two chain morphism $\mu_\bullet, \nu_\bullet : (M_\bullet, \partial_\bullet^M) \to (N_\bullet, \partial_\bullet^N)$ are called **homotopic**, and written $\mu_\bullet \sim \nu_\bullet$

Homotopy equivalences

Definition

- Two chain morphism $\mu_\bullet, \nu_\bullet : (M_\bullet, \partial_\bullet^M) \to (N_\bullet, \partial_\bullet^N)$ are called **homotopic**, and written $\mu_\bullet \sim \nu_\bullet$, if there exists a degree $+1$ chain morphism $h_\bullet : M_\bullet \to N_{\bullet+1}$ such that
$$\mu_\bullet - \nu_\bullet = \partial_\bullet^M h_\bullet + h_\bullet \partial_\bullet^N .$$

Homotopy equivalences

Definition

- Two chain morphism $\mu_\bullet, \nu_\bullet : (M_\bullet, \partial_\bullet^M) \to (N_\bullet, \partial_\bullet^N)$ are called **homotopic**, and written $\mu_\bullet \sim \nu_\bullet$, if there exists a degree $+1$ chain morphism $h_\bullet : M_\bullet \to N_{\bullet+1}$ such that
$$\mu_\bullet - \nu_\bullet = \partial_\bullet^M h_\bullet + h_\bullet \partial_\bullet^N.$$
- Two complexes are called **homotopy equivalent** if there exists chain morphisms $\mu_\bullet : M_\bullet \rightleftarrows N_\bullet : \nu_\bullet$ such that
$$\mu_\bullet \nu_\bullet \sim 1_\bullet^M : M_\bullet \to M_\bullet,$$
$$\nu_\bullet \mu_\bullet \sim 1_\bullet^N : N_\bullet \to N_\bullet.$$

Homotopy equivalences

Definition

- Two chain morphism $\mu_\bullet, \nu_\bullet : (M_\bullet, \partial_\bullet^M) \to (N_\bullet, \partial_\bullet^N)$ are called **homotopic**, and written $\mu_\bullet \sim \nu_\bullet$, if there exists a degree $+1$ chain morphism $h_\bullet : M_\bullet \to N_{\bullet+1}$ such that
$$\mu_\bullet - \nu_\bullet = \partial_\bullet^M h_\bullet + h_\bullet \partial_\bullet^N.$$
- Two complexes are called **homotopy equivalent** if there exists chain morphisms $\mu_\bullet : M_\bullet \rightleftarrows N_\bullet : \nu_\bullet$ such that
$$\mu_\bullet \nu_\bullet \sim 1_\bullet^M : M_\bullet \to M_\bullet,$$
$$\nu_\bullet \mu_\bullet \sim 1_\bullet^N : N_\bullet \to N_\bullet.$$

Corollary

Homotopy equivalent complexes are quasi-isomorphic.

Identify resolutions among each other

Let $\mathcal{A}$ be an ABELian category with enough projectives.

Identify resolutions among each other

Let $\mathcal{A}$ be an ABELian category with enough projectives.

Theorem

Any two projective resolutions in $\mathcal{A}$ are homotopy equivalent.

Identify resolutions among each other

Let $\mathcal{A}$ be an ABELian category with enough projectives.

Theorem

Any two projective resolutions in $\mathcal{A}$ are homotopy equivalent.

Definition

The **homotopy category** of $\mathcal{A}$ is defined as

$$\mathbf{K}(\mathcal{A}) := \mathbf{C}(\mathcal{A})/\text{homotopy equivalence.}$$

Identify resolutions among each other

Let $\mathcal{A}$ be an ABELian category with enough projectives.

Theorem

Any two projective resolutions in $\mathcal{A}$ are homotopy equivalent.

Definition

The **homotopy category** of $\mathcal{A}$ is defined as

$$\mathbf{K}(\mathcal{A}) := \mathbf{C}(\mathcal{A})/\text{homotopy equivalence.}$$

Corollary

Any two projective resolutions in $\mathcal{A}$ are isomorphic in $\mathbf{K}(\mathcal{A})$.

Identify resolutions among each other

Let $\mathcal{A}$ be an ABELian category with enough projectives.

Theorem

Any two projective resolutions in $\mathcal{A}$ are homotopy equivalent.

Definition

The **homotopy category** of $\mathcal{A}$ is defined as

$$\mathbf{K}(\mathcal{A}) := \mathbf{C}(\mathcal{A})/\text{homotopy equivalence.}$$

Corollary

Any two projective resolutions in $\mathcal{A}$ are isomorphic in $\mathbf{K}(\mathcal{A})$.

Problem

We still did not identify objects in $\mathcal{A}$ with their projective resolutions in $\mathbf{C}(\mathcal{A}) \supset \mathcal{A}$.

Two solutions to the last problem

Let $\mathcal{A}$ be an ABELian category with enough projectives

Two solutions to the last problem

Let $\mathcal{A}$ be an ABELian category with enough projectives and $\mathbf{P}(\mathcal{A}) \subset \mathbf{C}(\mathcal{A})$ the full subcategory of complexes with projective objects.

Two solutions to the last problem

Let $\mathcal{A}$ be an ABELian category with enough projectives and $\mathbf{P}(\mathcal{A}) \subset \mathbf{C}(\mathcal{A})$ the full subcategory of complexes with projective objects.

There are two solutions to the last problem:

1. **Restrict** $\mathbf{K}(\mathcal{A})$ to the full subcategory

$$\mathbf{K}(\mathbf{P}(\mathcal{A})) \subset \mathbf{K}(\mathcal{A}).$$

Two solutions to the last problem

Let $\mathcal{A}$ be an ABELian category with enough projectives and $\mathbf{P}(\mathcal{A}) \subset \mathbf{C}(\mathcal{A})$ the full subcategory of complexes with projective objects.

There are two solutions to the last problem:

1. **Restrict** $\mathbf{K}(\mathcal{A})$ to the full subcategory
$$\mathbf{K}(\mathbf{P}(\mathcal{A})) \subset \mathbf{K}(\mathcal{A}).$$
2. **Localize** $\mathbf{K}(\mathcal{A})$ at the class $\Sigma := \{\text{quasi-isomorphisms}\}$
$$\mathbf{D}(\mathcal{A}) := \Sigma^{-1}\mathbf{K}(\mathcal{A}).$$

Two solutions to the last problem

Let $\mathcal{A}$ be an ABELian category with enough projectives and $\mathbf{P}(\mathcal{A}) \subset \mathbf{C}(\mathcal{A})$ the full subcategory of complexes with projective objects.

There are two solutions to the last problem:

1. **Restrict** $\mathbf{K}(\mathcal{A})$ to the full subcategory
$$\mathbf{K}(\mathbf{P}(\mathcal{A})) \subset \mathbf{K}(\mathcal{A}).$$
2. **Localize** $\mathbf{K}(\mathcal{A})$ at the class $\Sigma := \{\text{quasi-isomorphisms}\}$
$$\mathbf{D}(\mathcal{A}) := \Sigma^{-1}\mathbf{K}(\mathcal{A}).$$

We call $\mathbf{D}(\mathcal{A})$ the **derived category** of $\mathcal{A}$.

Two solutions to the last problem

Let $\mathcal{A}$ be an ABELian category with enough projectives and $\mathbf{P}(\mathcal{A}) \subset \mathbf{C}(\mathcal{A})$ the full subcategory of complexes with projective objects.

There are two solutions to the last problem:

1. **Restrict** $\mathbf{K}(\mathcal{A})$ to the full subcategory
$$\mathbf{K}(\mathbf{P}(\mathcal{A})) \subset \mathbf{K}(\mathcal{A}).$$
2. **Localize** $\mathbf{K}(\mathcal{A})$ at the class $\Sigma := \{\text{quasi-isomorphisms}\}$
$$\mathbf{D}(\mathcal{A}) := \Sigma^{-1}\mathbf{K}(\mathcal{A}).$$
We call $\mathbf{D}(\mathcal{A})$ the **derived category** of $\mathcal{A}$.

Theorem

If $\mathcal{A}$ has enough projectives then the composition

$$\mathbf{K}(\mathbf{P}(\mathcal{A})) \hookrightarrow \mathbf{K}(\mathcal{A}) \to \mathbf{D}(\mathcal{A})$$

is an equivalence of categories.

The structure of a derived category

- The derived category $\mathbf{D}(\mathcal{A})$ is still additive, but generally not ABELian

The structure of a derived category

- The derived category $\mathbf{D}(\mathcal{A})$ is still additive, but generally not ABELian: it is a **triangulated** category.

The structure of a derived category

- The derived category $\mathbf{D}(\mathcal{A})$ is still additive, but generally not ABELian: it is a **triangulated** category.
- Although $\mathcal{A} \subset \mathbf{D}(\mathcal{A})$, there is in general no way to recover $\mathcal{A}$ from of $\mathbf{D}(\mathcal{A})$ equipped with its triangulated structure.

The structure of a derived category

- The derived category $\mathbf{D}(\mathcal{A})$ is still additive, but generally not ABELian: it is a **triangulated** category.
- Although $\mathcal{A} \subset \mathbf{D}(\mathcal{A})$, there is in general no way to recover $\mathcal{A}$ from of $\mathbf{D}(\mathcal{A})$ equipped with its triangulated structure.

Definition

Let us call a triangulated category $\mathcal{T}$ **nice** if there exists an ABELian category $\mathcal{A}$ with enough projectives such that

$$\mathcal{T} \simeq \mathbf{K}(\mathbf{P}(\mathcal{A})).$$

The structure of a derived category

- The derived category $\mathbf{D}(\mathcal{A})$ is still additive, but generally not ABELian: it is a **triangulated** category.
- Although $\mathcal{A} \subset \mathbf{D}(\mathcal{A})$, there is in general no way to recover $\mathcal{A}$ from of $\mathbf{D}(\mathcal{A})$ equipped with its triangulated structure.

Definition

Let us call a triangulated category $\mathcal{T}$ **nice** if there exists an ABELian category $\mathcal{A}$ with enough projectives such that

$$\mathcal{T} \simeq \mathbf{K}(\mathbf{P}(\mathcal{A})).$$

Corollary

If $\mathcal{A}$ has enough projectives then $\mathbf{D}(\mathcal{A})$ is nice.

The structure of a derived category

- The derived category $\mathbf{D}(\mathcal{A})$ is still additive, but generally not ABELian: it is a **triangulated** category.
- Although $\mathcal{A} \subset \mathbf{D}(\mathcal{A})$, there is in general no way to recover $\mathcal{A}$ from of $\mathbf{D}(\mathcal{A})$ equipped with its triangulated structure.

Definition

Let us call a triangulated category $\mathcal{T}$ **nice** if there exists an ABELian category $\mathcal{A}$ with enough projectives such that

$$\mathcal{T} \simeq \mathbf{K}(\mathbf{P}(\mathcal{A})).$$

Corollary

If $\mathcal{A}$ has enough projectives then $\mathbf{D}(\mathcal{A})$ is nice.

Corollary

Categories of modules over rings are nice.

What did we gain?

What did we gain by passing to the derived category $\mathbf{D}(\mathcal{A})$?

What did we gain?

What did we gain by passing to the derived category $\mathbf{D}(\mathcal{A})$?

- Identified each object with *all* its projective resolutions.

What did we gain?

What did we gain by passing to the derived category $\mathbf{D}(\mathcal{A})$?

- Identified each object with *all* its projective resolutions.
- $\operatorname{Ext}^k_{\mathcal{A}}(M,N) \cong \operatorname{Hom}_{\mathbf{D}(\mathcal{A})}(P^M_\bullet, P^N_{k+\bullet})$.

What did we gain?

What did we gain by passing to the derived category $\mathbf{D}(\mathcal{A})$?

- Identified each object with *all* its projective resolutions.
- $\mathrm{Ext}^k_{\mathcal{A}}(M,N) \cong \mathrm{Hom}_{\mathbf{D}(\mathcal{A})}(P^M_\bullet, P^N_{k+\bullet})$.
- ...

What did we gain?

What did we gain by passing to the derived category $\mathbf{D}(\mathcal{A})$?

- Identified each object with *all* its projective resolutions.
- $\mathrm{Ext}^k_{\mathcal{A}}(M, N) \cong \mathrm{Hom}_{\mathbf{D}(\mathcal{A})}(P^M_\bullet, P^N_{k+\bullet})$.
- ...
- $\mathbf{D}(\mathcal{A})$ might still be nice even if $\mathcal{A}$ does not have enough projectives

What did we gain?

What did we gain by passing to the derived category $\mathbf{D}(\mathcal{A})$?

- Identified each object with *all* its projective resolutions.
- $\mathrm{Ext}^k_{\mathcal{A}}(M, N) \cong \mathrm{Hom}_{\mathbf{D}(\mathcal{A})}(P^M_\bullet, P^N_{k+\bullet})$.
- ...
- $\mathbf{D}(\mathcal{A})$ might still be nice even if $\mathcal{A}$ does not have enough projectives, i.e.,

$$\mathbf{D}(\mathcal{A}) \simeq \mathbf{D}(\mathcal{B})$$

  where $\mathcal{B}$ is an ABELian category with enough projectives.

What did we gain?

What did we gain by passing to the derived category $\mathbf{D}(\mathcal{A})$?

- Identified each object with *all* its projective resolutions.
- $\mathrm{Ext}^k_{\mathcal{A}}(M,N) \cong \mathrm{Hom}_{\mathbf{D}(\mathcal{A})}(P^M_\bullet, P^N_{k+\bullet})$.
- ...
- $\mathbf{D}(\mathcal{A})$ might still be nice even if $\mathcal{A}$ does not have enough projectives, i.e.,

$$\mathbf{D}(\mathcal{A}) \simeq \mathbf{D}(\mathcal{B})$$

  where $\mathcal{B}$ is an ABELian category with enough projectives.

Definition

Two ABELian categories $\mathcal{A}$, $\mathcal{B}$ are called **derived equivalent** if

$$\mathbf{D}(\mathcal{A}) \simeq \mathbf{D}(\mathcal{B}).$$

Theorem (GROTHENDIECK)

The category of coherent sheaves $\mathfrak{Coh}\,\mathbb{P}^n$ does not have enough projectives.

Theorem (GROTHENDIECK)

The category of coherent sheaves $\mathfrak{Coh}\,\mathbb{P}^n$ does not have enough projectives.

Still, the category $\mathfrak{Coh}\,(\mathbb{P}^n)$ is nice

A wormhole between geometry and algebra

Theorem (Grothendieck)

The category of coherent sheaves $\mathfrak{Coh}\, \mathbb{P}^n$ does not have enough projectives.

Still, the category $\mathfrak{Coh}\,(\mathbb{P}^n)$ is nice:

Theorem (Beilinson)

The category $\mathfrak{Coh}\, \mathbb{P}^n$ admits a ***tilting object***

A wormhole between geometry and algebra

Theorem (Grothendieck)

The category of coherent sheaves $\mathfrak{Coh}\,\mathbb{P}^n$ does not have enough projectives.

Still, the category $\mathfrak{Coh}\,(\mathbb{P}^n)$ is nice:

Theorem (Beilinson)

*The category $\mathfrak{Coh}\,\mathbb{P}^n$ admits a **tilting object**, i.e., an object $T \in \mathfrak{Coh}\,\mathbb{P}^n$ such that*

$$T \otimes_{\mathrm{End}(T)} - : \mathbf{D}(\mathrm{End}(T)\text{-}mod) \to \mathbf{D}(\mathfrak{Coh}\,\mathbb{P}^n)$$

is a triangulated equivalence of categories.

A wormhole between geometry and algebra

Theorem (Grothendieck)

The category of coherent sheaves $\mathfrak{Coh}\,\mathbb{P}^n$ does not have enough projectives.

Still, the category $\mathfrak{Coh}\,(\mathbb{P}^n)$ is nice:

Theorem (Beilinson)

The category $\mathfrak{Coh}\,\mathbb{P}^n$ admits a ***tilting object****, i.e., an object $T \in \mathfrak{Coh}\,\mathbb{P}^n$ such that*

$$T \otimes_{\mathrm{End}(T)} - : \mathbf{D}(\mathrm{End}(T)\text{-}mod) \to \mathbf{D}(\mathfrak{Coh}\,\mathbb{P}^n)$$

is a triangulated equivalence of categories.

This derived equivalence is a wormhole between the **algebraic geometry** of $\mathbb{P}^n$ and the **representation theory** of the *finite dimensional* algeba $\mathrm{End}(T)$.

Mathematical wormholes

Derived equivalences are wormholes in the universe of mathematics, able to connect seemingly remote fields:

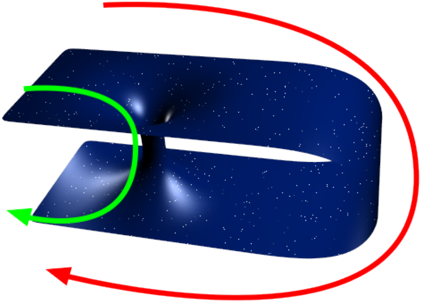


Figure: License: GNU-FDL, made by Panzi

Why implement categories on the computer?

Why do we need to implement categories on the computer?

Why implement categories on the computer?

Why do we need to implement categories on the computer?

- They are a very flexible modeling tool which helps us making highly **abstract** mathematics **constructive**.

Why implement categories on the computer?

Why do we need to implement categories on the computer?

- They are a very flexible modeling tool which helps us making highly **abstract** mathematics **constructive**.
- They relate seemingly remote fields of mathematics.

Why implement categories on the computer?

Why do we need to implement categories on the computer?

- They are a very flexible modeling tool which helps us making highly **abstract** mathematics **constructive**.
- They relate seemingly remote fields of mathematics.
- The latter has an invaluable advantage for **algorithmic** mathematics as we can often enough use (derived) equivalences to

Why do we need to implement categories on the computer?

- They are a very flexible modeling tool which helps us making highly **abstract** mathematics **constructive**.
- They relate seemingly remote fields of mathematics.
- The latter has an invaluable advantage for **algorithmic** mathematics as we can often enough use (derived) equivalences to
  1. pass to more efficient data structures

Why implement categories on the computer?

Why do we need to implement categories on the computer?

- They are a very flexible modeling tool which helps us making highly **abstract** mathematics **constructive**.
- They relate seemingly remote fields of mathematics.
- The latter has an invaluable advantage for **algorithmic** mathematics as we can often enough use (derived) equivalences to
  1. pass to more efficient data structures and
  2. translate computational contexts in which runtime complexities of algorithms are **exponential**

Why implement categories on the computer?

Why do we need to implement categories on the computer?

- They are a very flexible modeling tool which helps us making highly **abstract** mathematics **constructive**.
- They relate seemingly remote fields of mathematics.
- The latter has an invaluable advantage for **algorithmic** mathematics as we can often enough use (derived) equivalences to
  1. pass to more efficient data structures and
  2. translate computational contexts in which runtime complexities of algorithms are **exponential** to contexts in which the corresponding algorithm have **polynomial** runtime complexity!

Why implement categories on the computer?

Why do we need to implement categories on the computer?

- They are a very flexible modeling tool which helps us making highly **abstract** mathematics **constructive**.
- They relate seemingly remote fields of mathematics.
- The latter has an invaluable advantage for **algorithmic** mathematics as we can often enough use (derived) equivalences to
  1. pass to more efficient data structures and
  2. translate computational contexts in which runtime complexities of algorithms are **exponential** to contexts in which the corresponding algorithm have **polynomial** runtime complexity!

These are the reasons why we are so eager to build software helping us to travel through mathematical wormholes.

Thank you

Mohamed Barakat and Markus Lange-Hegermann, *An axiomatic setup for algorithmic homological algebra and an alternative approach to localization*, J. Algebra Appl. **10** (2011), no. 2, 269–293, (`arXiv:1003.1943`). MR 2795737 (2012f:18022)

______, *Gabriel morphisms and the computability of Serre quotients with applications to coherent sheaves*, (`arXiv:1409.2028`), 2014.