Constructive Category Theory and Applications in Algebraic Geometry

Sebastian Gutsche

Universität Siegen

Siegen, August 31, 2017

Outline

Outline

Constructive category theory

Abstraction of language

Abstraction of language

Addition of two numbers:

Data type: `int`

Data type: `float`

Abstraction of language

Addition of two numbers: Assembly

Data type: `int`

Data type: `float`

Abstraction of language

Addition of two numbers: Assembly

Data type: `int`

```
addi:
movl   %edi, -4(%rsp)
movl   %esi, -8(%rsp)
movl   -4(%rsp), %esi
addl   -8(%rsp), %esi
movl   %esi, %eax
ret
```

Data type: `float`

Abstraction of language

Addition of two numbers: Assembly

Data type: `int`

```
addi:
movl  %edi, -4(%rsp)
movl  %esi, -8(%rsp)
movl  -4(%rsp), %esi
addl  -8(%rsp), %esi
movl  %esi, %eax
ret
```

Data type: `float`

```
addf:
movss %xmm0, -4(%rsp)
movss %xmm1, -8(%rsp)
movss -4(%rsp), %xmm0
addss -8(%rsp), %xmm0
ret
```

Abstraction of language

Addition of two numbers: C

Data type: `int`

Data type: `float`

Abstraction of language

Addition of two numbers: C

Data type: `int`

```
int addi( int a,
          int b )
{
  return a + b;
}
```

Data type: `float`

Abstraction of language

Addition of two numbers: C

Data type: `int`

```
int addi( int a,
          int b )
{
  return a + b;
}
```

Data type: `float`

```
float addf( float a,
            float b )
{
  return a + b;
}
```

Abstraction of language

Addition of two numbers: GAP or Julia

Data type: `int`

Data type: `float`

Abstraction of language

Addition of two numbers: GAP or Julia

Data type: `int`

```
function( a, b )
    return a + b;
end;
```

Data type: `float`

Abstraction of language

Addition of two numbers: GAP or Julia

Data type: `int`

```
function( a, b )
    return a + b;
end;
```

Data type: `float`

```
function( a, b )
    return a + b;
end;
```

Abstraction of language

Addition of two numbers: GAP or Julia

Data type: `int, float`

```
function( a, b )
    return a + b;
end;
```

Abstraction of language

Addition of two numbers: GAP or Julia

Data type: `int, float`

```
function( a, b )
    return a + b;
end;
```

High language leads to generic code!

Abstraction of language

Computing the intersection of two subobjects

Abstraction of language

Computing the intersection of two subobjects

Vector spaces

$\langle v_1, v_2 \rangle, \langle w_1, w_2 \rangle \leq V$:

Abstraction of language

Computing the intersection of two subobjects

Vector spaces

$\langle v_1, v_2 \rangle, \langle w_1, w_2 \rangle \leq V$:
Solution of

$$\begin{aligned} & x_1 v_1 + x_2 v_2 \\ = & y_1 w_1 + y_2 w_2 \end{aligned}$$

Abstraction of language

Computing the intersection of two subobjects

Vector spaces

$\langle v_1, v_2 \rangle, \langle w_1, w_2 \rangle \leq V$:
Solution of

$$\begin{aligned} & x_1 v_1 + x_2 v_2 \\ = & y_1 w_1 + y_2 w_2 \end{aligned}$$

Ideals of $\mathbb{Z}$

Abstraction of language

Computing the intersection of two subobjects

Vector spaces

$\langle v_1, v_2 \rangle, \langle w_1, w_2 \rangle \leq V$:
Solution of

$$\begin{aligned} & x_1 v_1 + x_2 v_2 \\ = & y_1 w_1 + y_2 w_2 \end{aligned}$$

Ideals of $\mathbb{Z}$

$\langle x \rangle, \langle y \rangle \leq \mathbb{Z}$:

Abstraction of language

Computing the intersection of two subobjects

Vector spaces

$\langle v_1, v_2 \rangle, \langle w_1, w_2 \rangle \leq V$:
Solution of

$$\begin{aligned} & x_1 v_1 + x_2 v_2 \\ = & y_1 w_1 + y_2 w_2 \end{aligned}$$

Ideals of $\mathbb{Z}$

$\langle x \rangle, \langle y \rangle \leq \mathbb{Z}$:
Euclidean algorithm:

$$\langle \mathrm{lcm}\,(x, y) \rangle$$

Abstraction of language

Computing the intersection of two subobjects

Vector spaces

$\langle v_1, v_2 \rangle, \langle w_1, w_2 \rangle \leq V$:
Solution of

$$\begin{aligned} & x_1 v_1 + x_2 v_2 \\ = & y_1 w_1 + y_2 w_2 \end{aligned}$$

Ideals of $\mathbb{Z}$

$\langle x \rangle, \langle y \rangle \leq \mathbb{Z}$:
Euclidean algorithm:

$$\langle \mathrm{lcm}(x, y) \rangle$$

Generic algorithm for both cases?

Abstraction of language

Computing the intersection of two subobjects

Vector spaces

$\langle v_1, v_2 \rangle, \langle w_1, w_2 \rangle \leq V$:
Solution of

$$\begin{aligned} & x_1 v_1 + x_2 v_2 \\ = & y_1 w_1 + y_2 w_2 \end{aligned}$$

Ideals of $\mathbb{Z}$

$\langle x \rangle, \langle y \rangle \leq \mathbb{Z}$:
Euclidean algorithm:

$$\langle \operatorname{lcm}(x, y) \rangle$$

Generic algorithm for both cases? **Category theory!**

Category theory as programming language

Category theory

Category theory as programming language

Category theory

- abstracts mathematical structures

Category theory as programming language

Category theory

- abstracts mathematical structures
- defines a *language* to formulate theorems and algorithms for different structures *at the same time*

Category theory as programming language

Category theory

- abstracts mathematical structures
- defines a *language* to formulate theorems and algorithms for different structures *at the same time*

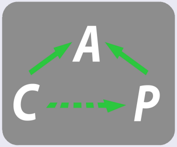
CAP - Categories, Algorithms, and Programming

Category theory as programming language

Category theory

- abstracts mathematical structures
- defines a *language* to formulate theorems and algorithms for different structures *at the same time*

CAP - Categories, Algorithms, and Programming



CAP implements a
categorical programming language

Categories

Definition

A category $\mathcal{A}$ contains the following data:

Categories

Definition

A category $\mathcal{A}$ contains the following data:

- $\mathrm{Obj}_{\mathcal{A}}$

$A \qquad B \qquad C$

Categories

Definition

A category $\mathcal{A}$ contains the following data:

- $\mathrm{Obj}_{\mathcal{A}}$
- $\mathrm{Hom}_{\mathcal{A}}(A, B)$

$$A \qquad\qquad B \qquad\qquad C$$

Categories

Definition

A category $\mathcal{A}$ contains the following data:

- $\mathrm{Obj}_{\mathcal{A}}$
- $\mathrm{Hom}_{\mathcal{A}}(A, B)$

$$A \longrightarrow B \qquad\qquad C$$

Categories

Definition

A category $\mathcal{A}$ contains the following data:

- $\mathrm{Obj}_{\mathcal{A}}$
- $\mathrm{Hom}_{\mathcal{A}}(A, B)$

$$A \longrightarrow B \longrightarrow C$$

Categories

Definition

A category $\mathcal{A}$ contains the following data:

- $\mathrm{Obj}_{\mathcal{A}}$
- $\mathrm{Hom}_{\mathcal{A}}(A, B)$
- $\circ : \mathrm{Hom}_{\mathcal{A}}(B, C) \times \mathrm{Hom}_{\mathcal{A}}(A, B) \to \mathrm{Hom}_{\mathcal{A}}(A, C)$ (assoz.)

$$A \longrightarrow B \longrightarrow C$$

Categories

Definition

A category $\mathcal{A}$ contains the following data:

- $\mathrm{Obj}_{\mathcal{A}}$
- $\mathrm{Hom}_{\mathcal{A}}(A, B)$
- $\circ : \mathrm{Hom}_{\mathcal{A}}(B, C) \times \mathrm{Hom}_{\mathcal{A}}(A, B) \to \mathrm{Hom}_{\mathcal{A}}(A, C) \quad$ (assoz.)

$$\begin{tikzcd} A \arrow[r] \arrow[rr, bend left] & B \arrow[r] & C \end{tikzcd}$$

Categories

Definition

A category $\mathcal{A}$ contains the following data:

- $\mathrm{Obj}_{\mathcal{A}}$
- $\mathrm{Hom}_{\mathcal{A}}(A, B)$
- $\circ : \mathrm{Hom}_{\mathcal{A}}(B, C) \times \mathrm{Hom}_{\mathcal{A}}(A, B) \to \mathrm{Hom}_{\mathcal{A}}(A, C)$ (assoz.)
- Neutral elements: $\mathrm{id}_A \in \mathrm{Hom}_{\mathcal{A}}(A, A)$

$$\begin{tikzcd} A \arrow[r] \arrow[rr, bend left] & B \arrow[r] & C \end{tikzcd}$$

Categories

Definition

A category $\mathcal{A}$ contains the following data:

- $\mathrm{Obj}_{\mathcal{A}}$
- $\mathrm{Hom}_{\mathcal{A}}(A, B)$
- $\circ : \mathrm{Hom}_{\mathcal{A}}(B, C) \times \mathrm{Hom}_{\mathcal{A}}(A, B) \to \mathrm{Hom}_{\mathcal{A}}(A, C)$ (assoz.)
- Neutral elements: $\mathrm{id}_A \in \mathrm{Hom}_{\mathcal{A}}(A, A)$

$$\begin{tikzcd} A \arrow[r] \arrow[rr, bend left] \arrow[loop below] & B \arrow[r] \arrow[loop below] & C \arrow[loop below] \end{tikzcd}$$

Categories

Definition

A category $\mathcal{A}$ contains the following data:

- $\mathrm{Obj}_{\mathcal{A}}$
- $\mathrm{Hom}_{\mathcal{A}}(A, B)$
- $\circ : \mathrm{Hom}_{\mathcal{A}}(B, C) \times \mathrm{Hom}_{\mathcal{A}}(A, B) \to \mathrm{Hom}_{\mathcal{A}}(A, C)$ (assoz.)
- Neutral elements: $\mathrm{id}_A \in \mathrm{Hom}_{\mathcal{A}}(A, A)$

$$\begin{tikzcd} A \arrow[r] \arrow[rr, bend left] \arrow[loop below] & B \arrow[r] \arrow[loop below] & C \arrow[loop below] \end{tikzcd}$$

Computable categories

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

Finitely generated $\mathbb{Q}$-vector spaces (skeletal)

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

Finitely generated $\mathbb{Q}$-vector spaces (skeletal)

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

Finitely generated $\mathbb{Q}$-vector spaces (skeletal)

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

Finitely generated $\mathbb{Q}$-vector spaces (skeletal)

$1 \qquad 2 \qquad 1$

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

Finitely generated $\mathbb{Q}$-vector spaces (skeletal)

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

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

Finitely generated $\mathbb{Q}$-vector spaces (skeletal)

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

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

Finitely generated $\mathbb{Q}$-vector spaces (skeletal)

$$\begin{tikzcd} 1 \arrow[r, "\begin{pmatrix} 1 & 2 \end{pmatrix}"'] \arrow[rr, bend left, "\begin{pmatrix} 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 4 \end{pmatrix} = (11)"] & 2 \arrow[r, "\begin{pmatrix} 3 \\ 4 \end{pmatrix}"'] & 1 \end{tikzcd}$$

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

Finitely generated $\mathbb{Q}$-vector spaces (skeletal)

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

$$\begin{tikzcd} 1 \arrow[r, "\begin{pmatrix} 1 & 2 \end{pmatrix}"'] \arrow[rr, bend left] & 2 \arrow[r, "\begin{pmatrix} 3 \\ 4 \end{pmatrix}"'] & 1 \end{tikzcd}$$

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

Finitely generated $\mathbb{Q}$-vector spaces (skeletal)

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

$$\begin{tikzcd}
1 \arrow[r, "\begin{pmatrix} 1 & 2 \end{pmatrix}"'] \arrow[rr, bend left] \arrow[loop below, "(1)"] & 2 \arrow[r, "\begin{pmatrix} 3 \\ 4 \end{pmatrix}"'] \arrow[loop below, "\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}"] & 1 \arrow[loop below, "(1)"]
\end{tikzcd}$$

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

Finitely generated $\mathbb{Q}$-vector spaces (skeletal)

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

$$\begin{tikzcd}
1 \arrow[r, "\begin{pmatrix} 1 & 2 \end{pmatrix}"'] \arrow[rr, bend left] \arrow[loop below, "(1)"] & 2 \arrow[r, "\begin{pmatrix} 3 \\ 4 \end{pmatrix}"'] \arrow[loop below, "\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}"] & 1 \arrow[loop below, "(1)"]
\end{tikzcd}$$

Categorical operations

Some categorical operations in abelian categories

Categorical operations

Some categorical operations in abelian categories

- Zero morphisms

Categorical operations

Some categorical operations in abelian categories

- Zero morphisms
- Addition and subtraction of morphisms

Categorical operations

Some categorical operations in abelian categories

- Zero morphisms
- Addition and subtraction of morphisms
- Direct sums

Categorical operations

Some categorical operations in abelian categories

- Zero morphisms
- Addition and subtraction of morphisms
- Direct sums
- Kernels and cokernels of morphisms

Categorical operations

Some categorical operations in abelian categories

- Zero morphisms
- Addition and subtraction of morphisms
- Direct sums
- Kernels and cokernels of morphisms
- ...

Implementation of the kernel

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

Implementation of the kernel

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

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

Implementation of the kernel

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

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

Implementation of the kernel

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

... one needs an object $\ker \varphi$,

$$\begin{array}{ccc} \ker \varphi & & \\ & A \xrightarrow{\varphi} B \end{array}$$

Implementation of the kernel

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

... one needs an object $\ker \varphi$,
its embedding $\kappa = \mathrm{KernelEmbedding}(\varphi)$,

$$\begin{tikzcd} \ker \varphi \arrow[dr, hook, "\kappa"] & & \\ & A \arrow[r, "\varphi"] & B \end{tikzcd}$$

Implementation of the kernel

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

... one needs an object $\ker\varphi$,
its embedding $\kappa = \mathrm{KernelEmbedding}(\varphi)$,
and for every test morphism τ

$$\begin{tikzcd}
\ker\varphi \arrow[dr, hook, "\kappa"] \arrow[drr, bend left, "0"] & & \\
& A \arrow[r, "\varphi"] & B \\
T \arrow[ur, "\tau"] \arrow[urr, bend right, "0"'] & &
\end{tikzcd}$$

Implementation of the kernel

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

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

$$
\begin{tikzcd}
\ker\varphi \arrow[dr, hook, "\kappa"] \arrow[drr, bend left, "0"] & & \\
 & A \arrow[r, "\varphi"] & B \\
T \arrow[uu, dashed, "\lambda"] \arrow[ur, "\tau"] \arrow[urr, bend right, "0"'] & &
\end{tikzcd}
$$

Implementation of the kernel

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

... one needs an object $\ker \varphi$,
its embedding $\kappa = \mathrm{KernelEmbedding}(\varphi)$,
and for every test morphism τ
a *unique* morphism $\lambda = \mathrm{KernelLift}(\varphi, \tau)$, such that

$$
\begin{tikzcd}
\ker\varphi \arrow[dr, hook, "\kappa"] \arrow[drr, bend left, "0"] & & \\
 & A \arrow[r, "\varphi"] & B \\
T \arrow[uu, dashed, "\lambda"] \arrow[ur, "\tau"'] \arrow[urr, bend right, "0"'] \arrow[u, phantom, "\circlearrowright"]
\end{tikzcd}
$$

Implementation of the kernel: $\mathbb{Q}$-vector spaces

$\mathrm{Obj} := \mathbb{Z}_{\geq 0}$, $\mathrm{Hom}\,(m, n) := \mathbb{Q}^{m \times n}$

Implementation of the kernel: $\mathbb{Q}$-vector spaces

$\mathrm{Obj} := \mathbb{Z}_{\geq 0}$, $\mathrm{Hom}\,(m, n) := \mathbb{Q}^{m \times n}$

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

Implementation of the kernel: $\mathbb{Q}$-vector spaces

$\mathrm{Obj} := \mathbb{Z}_{\geq 0}$, $\mathrm{Hom}\,(m, n) := \mathbb{Q}^{m \times n}$

$\ker \varphi$

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

Implementation of the kernel: $\mathbb{Q}$-vector spaces

$\mathrm{Obj} := \mathbb{Z}_{\geq 0}$, $\mathrm{Hom}\,(m,n) := \mathbb{Q}^{m \times n}$

$\ker \varphi$

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

Compute

- $\ker \varphi$ as $\dim(A) - \mathrm{rank}(\varphi)$

Implementation of the kernel: $\mathbb{Q}$-vector spaces

$\mathrm{Obj} := \mathbb{Z}_{\geq 0}$, $\mathrm{Hom}\,(m, n) := \mathbb{Q}^{m \times n}$

$$\begin{tikzcd} \ker \varphi \arrow[dr, hook, "\kappa"] & & \\ & A \arrow[r, "\varphi"] & B \end{tikzcd}$$

Compute

- $\ker \varphi$ as $\mathrm{dim}(A) - \mathrm{rank}(\varphi)$

Implementation of the kernel: $\mathbb{Q}$-vector spaces

$\mathrm{Obj} := \mathbb{Z}_{\geq 0}$, Hom $(m, n) := \mathbb{Q}^{m \times n}$

$$\begin{tikzcd}
\ker \varphi \arrow[dr, hook, "\kappa"] & & \\
& A \arrow[r, "\varphi"] & B
\end{tikzcd}$$

Compute

- $\ker \varphi$ as $\dim(A) - \mathrm{rank}(\varphi)$
- κ by solving $X \cdot \varphi = 0$

Implementation of the kernel: $\mathbb{Q}$-vector spaces

$\mathrm{Obj} := \mathbb{Z}_{\geq 0}$, $\mathrm{Hom}\,(m, n) := \mathbb{Q}^{m \times n}$

$$\begin{tikzcd}
\ker \varphi \arrow[dr, hook, "\kappa"] & & \\
& A \arrow[r, "\varphi"] & B \\
T \arrow[ur, "\tau"] & &
\end{tikzcd}$$

Compute

- $\ker \varphi$ as $\dim(A) - \mathrm{rank}(\varphi)$
- κ by solving $X \cdot \varphi = 0$

Implementation of the kernel: $\mathbb{Q}$-vector spaces

$\mathrm{Obj} := \mathbb{Z}_{\geq 0}$, $\mathrm{Hom}\,(m, n) := \mathbb{Q}^{m \times n}$

$$\begin{tikzcd}
\ker\varphi \arrow[dr, hook, "\kappa"] & & \\
 & A \arrow[r, "\varphi"] & B \\
T \arrow[uu, dashed, "\lambda"] \arrow[ur, "\tau"'] & &
\end{tikzcd}$$

Compute

- $\ker\varphi$ as $\dim(A) - \mathrm{rank}(\varphi)$
- κ by solving $X \cdot \varphi = 0$

Implementation of the kernel: $\mathbb{Q}$-vector spaces

$\mathrm{Obj} := \mathbb{Z}_{\geq 0}$, $\mathrm{Hom}\,(m, n) := \mathbb{Q}^{m \times n}$

$$\begin{tikzcd} \ker\varphi \arrow[rr, hook, "\kappa"] & & A \arrow[r, "\varphi"] & B \\ T \arrow[u, dashed, "\lambda"] \arrow[urr, "\tau"'] & & & \end{tikzcd}$$

Compute

- $\ker\varphi$ as $\dim(A) - \mathrm{rank}(\varphi)$
- κ by solving $X \cdot \varphi = 0$
- λ by solving $X \cdot \kappa = \tau$

What is CAP?

CAP - Categories, Algorithms, and Programming

What is CAP?

CAP - Categories, Algorithms, and Programming

CAP is a framework to implement computable categories and provides

What is CAP?

CAP - Categories, Algorithms, and Programming

CAP is a framework to implement computable categories and provides

- specifications of categorical operations,

What is CAP?

CAP - Categories, Algorithms, and Programming

CAP is a framework to implement computable categories and provides

- specifications of categorical operations,
- generic algorithms based on basic categorical operations,

What is CAP?

CAP - Categories, Algorithms, and Programming

CAP is a framework to implement computable categories and provides

- specifications of categorical operations,
- generic algorithms based on basic categorical operations,
- a categorical programming language having categorical operations as syntax elements.

Computing the intersection

Let $M_1 \subseteq N$ and $M_2 \subseteq N$ subobjects.

Computing the intersection

Let $M_1 \hookrightarrow N$ and $M_2 \hookrightarrow N$ subobjects.

Computing the intersection

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

$$\begin{tikzcd}
M_1 \arrow[dr, hook, "\iota_1"] & \\
& N \\
M_2 \arrow[ur, hook, "\iota_2"'] &
\end{tikzcd}$$

Computing the intersection

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

$$\begin{tikzcd}
 & & M_1 \arrow[dr, hook, "\iota_1"] & \\
M_1 \oplus M_2 & & & N \\
 & & M_2 \arrow[ur, hook, "\iota_2"'] &
\end{tikzcd}$$

Computing the intersection

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

$$\begin{tikzcd}
 & M_1 \arrow[dr, hook, "\iota_1"] & \\
M_1 \oplus M_2 \arrow[ur, two heads, "\pi_1"] & & N \\
 & M_2 \arrow[ur, hook, "\iota_2"'] &
\end{tikzcd}$$

Computing the intersection

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

$$\begin{tikzcd}
 & M_1 \arrow[dr, hook, "\iota_1"] & \\
M_1 \oplus M_2 \arrow[ur, two heads, "\pi_1"] \arrow[dr, two heads, "\pi_2"'] & & N \\
 & M_2 \arrow[ur, hook, "\iota_2"'] &
\end{tikzcd}$$

Computing the intersection

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

$$\begin{tikzcd} & M_1 \arrow[dr, hook, "\iota_1"] & \\ M_1 \oplus M_2 \arrow[ur, two heads, "\pi_1"] \arrow[dr, two heads, "\pi_2"'] & & N \\ & M_2 \arrow[ur, hook, "\iota_2"'] & \end{tikzcd}$$

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

Computing the intersection

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

$$\begin{tikzcd}
 & M_1 \arrow[dr, hook, "\iota_1"] & \\
M_1 \oplus M_2 \arrow[ur, two heads, "\pi_1"] \arrow[rr, "\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2"] \arrow[dr, two heads, "\pi_2"'] & & N \\
 & M_2 \arrow[ur, hook, "\iota_2"'] &
\end{tikzcd}$$

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

Computing the intersection

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

$$\begin{tikzcd}
 & M_1 \arrow[dr, hook, "\iota_1"] & \\
M_1 \oplus M_2 \arrow[ur, two heads, "\pi_1"] \arrow[dr, two heads, "\pi_2"'] \arrow[rr, "\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2"] & & N \\
 & M_2 \arrow[ur, hook, "\iota_2"'] &
\end{tikzcd}$$

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

Computing the intersection

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

$$\begin{tikzcd}
 & & M_1 \arrow[dr, hook, "\iota_1"] & \\
M_1 \cap M_2 \arrow[r, hook, "\kappa"] & M_1 \oplus M_2 \arrow[ur, two heads, "\pi_1"] \arrow[dr, two heads, "\pi_2"'] \arrow[rr, "\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2"] & & N \\
 & & M_2 \arrow[ur, hook, "\iota_2"'] &
\end{tikzcd}$$

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

Computing the intersection

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

$$\begin{tikzcd}
 & & M_1 \arrow[dr, hook, "\iota_1"] & \\
M_1 \cap M_2 \arrow[r, hook, "\kappa"] & M_1 \oplus M_2 \arrow[ur, two heads, "\pi_1"] \arrow[dr, two heads, "\pi_2"'] \arrow[rr, "\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2"] & & N \\
 & & M_2 \arrow[ur, hook, "\iota_2"'] &
\end{tikzcd}$$

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

Computing the intersection

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

$$\begin{tikzcd}
 & & M_1 \arrow[dr, hook, "\iota_1"] & \\
M_1 \cap M_2 \arrow[r, hook, "\kappa"] & M_1 \oplus M_2 \arrow[ur, two heads, "\pi_1"] \arrow[dr, two heads, "\pi_2"'] \arrow[rr, "\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2"] & & N \\
 & & M_2 \arrow[ur, hook, "\iota_2"'] &
\end{tikzcd}$$

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

Computing the intersection

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

$$\begin{tikzcd}
 & & M_1 \arrow[dr, hook, "\iota_1"] & \\
M_1 \cap M_2 \arrow[r, hook, "\kappa"] & M_1 \oplus M_2 \arrow[ur, two heads, "\pi_1"] \arrow[rr, "\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2"] \arrow[dr, two heads, "\pi_2"'] & & N \\
 & & M_2 \arrow[ur, hook, "\iota_2"'] &
\end{tikzcd}$$

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

Translation to CAP

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

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

$$\kappa := \mathrm{KernelEmbedding}\left(\varphi\right)$$

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

Translation to CAP

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

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

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

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

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

Translation to CAP

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

```
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 );
```

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

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

Translation to CAP

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

```
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 );
```

$\kappa := \mathrm{KernelEmbedding}\left(\varphi\right)$

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

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

Translation to CAP

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

```
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 );
```

$\kappa := \mathrm{KernelEmbedding}\left(\varphi\right)$

```
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

```
Schnitt := 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

```
Schnitt := 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

```
Schnitt := 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

```
Schnitt := 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: $\mathbb{Q}$-vector space

Compute the intersection of

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

Computing the intersection: $\mathbb{Q}$-vector space

Compute the intersection of

$$
\begin{tikzcd}
M_1 \arrow[r, hook, "\iota_1 := \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}"] \arrow[d, phantom, "\shortparallel"] & N \arrow[d, phantom, "\shortparallel"] & M_2 \arrow[l, hook', "\iota_2 := \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}"'] \arrow[d, phantom, "\shortparallel"] \\
2 & 3 & 2
\end{tikzcd}
$$

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

Computing the intersection: $\mathbb{Q}$-vector space

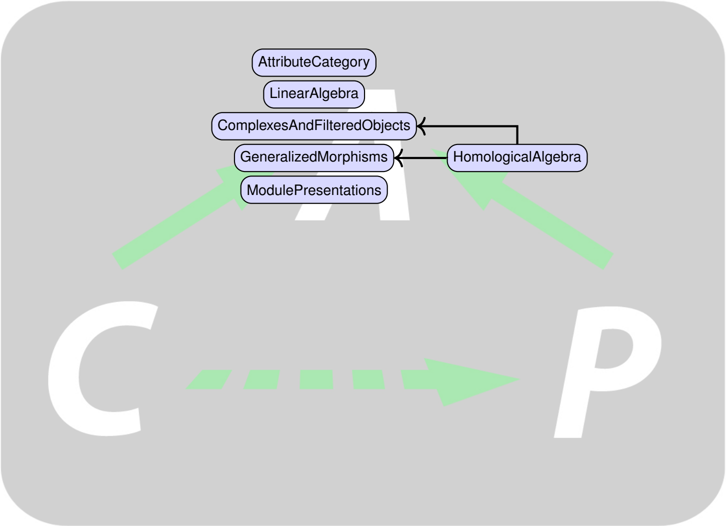
Compute the intersection of

$$\begin{tikzcd} M_1 \arrow[r, hook, "\iota_1 := \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}"] \arrow[d, phantom, "\shortparallel"] & N \arrow[d, phantom, "\shortparallel"] & M_2 \arrow[l, hook', "\iota_2 := \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}"'] \arrow[d, phantom, "\shortparallel"] \\ 2 & 3 & 2 \end{tikzcd}$$

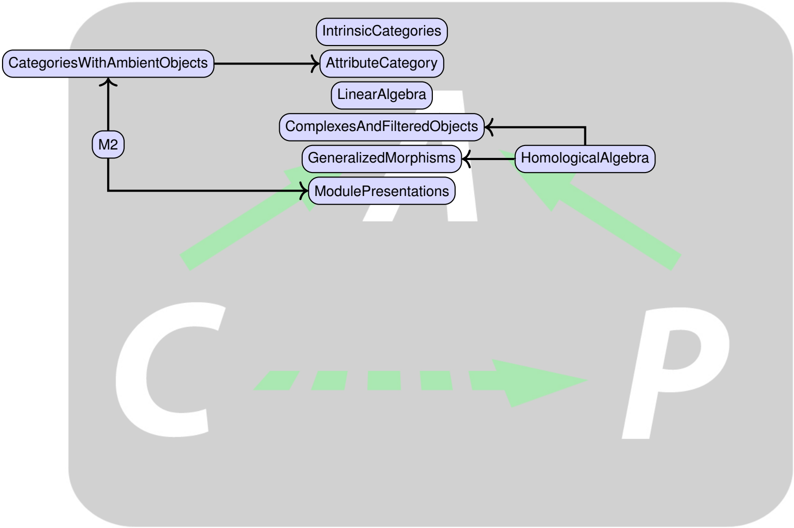
```
gap> gamma := Schnitt( iota1, iota2 );
<A morphism in the category of matrices over Q>
gap> Display( gamma );
[ [  1,  1,  0 ] ]

A morphism in the category of matrices over Q
```

CAP packages

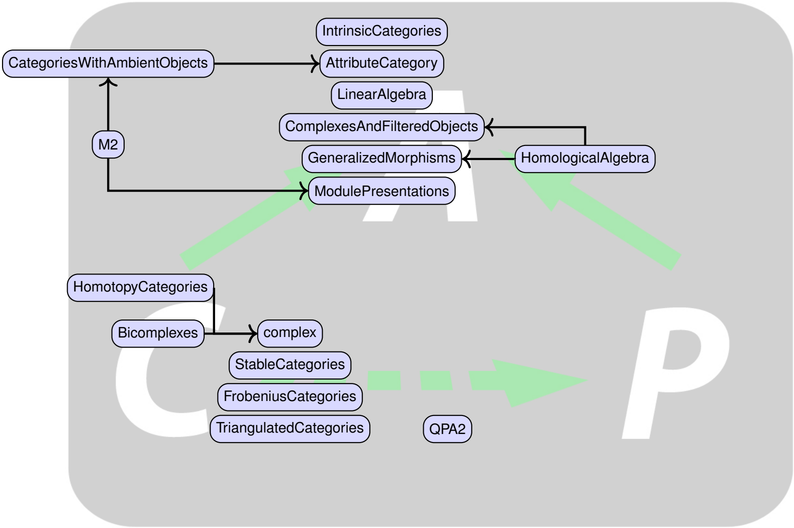
CAP packages



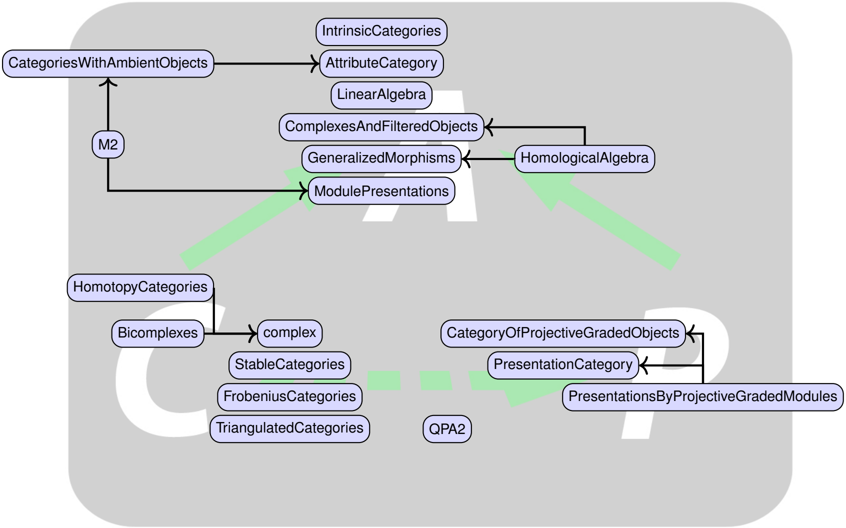
CAP packages



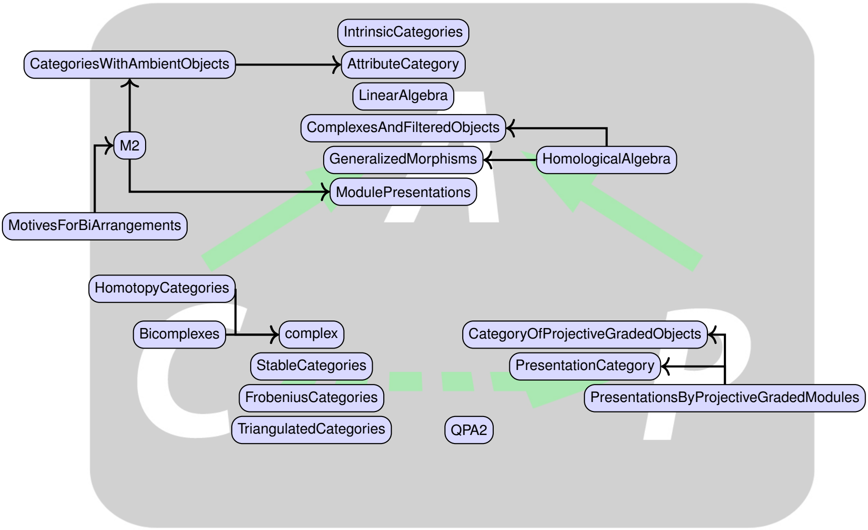
CAP packages



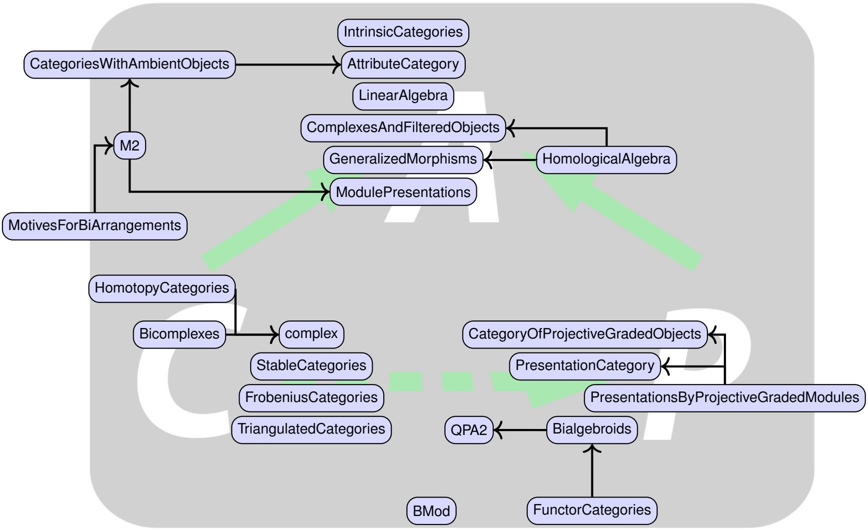
CAP packages



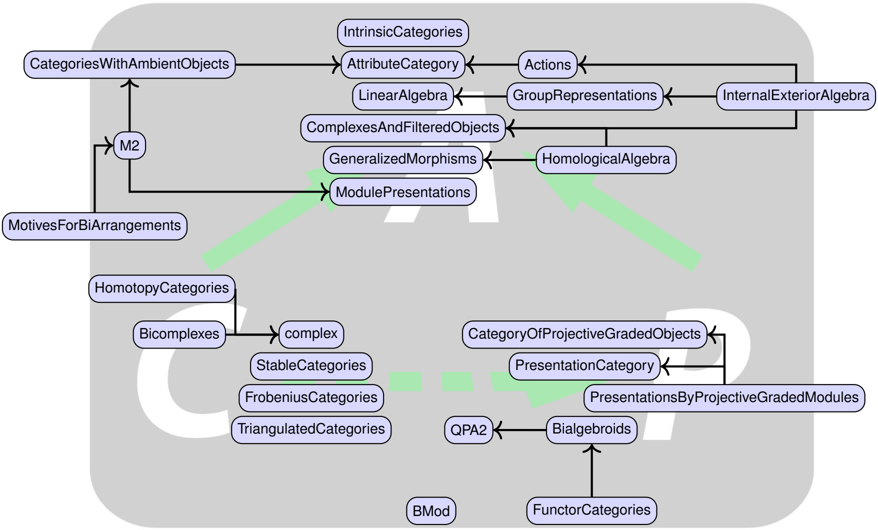
CAP packages



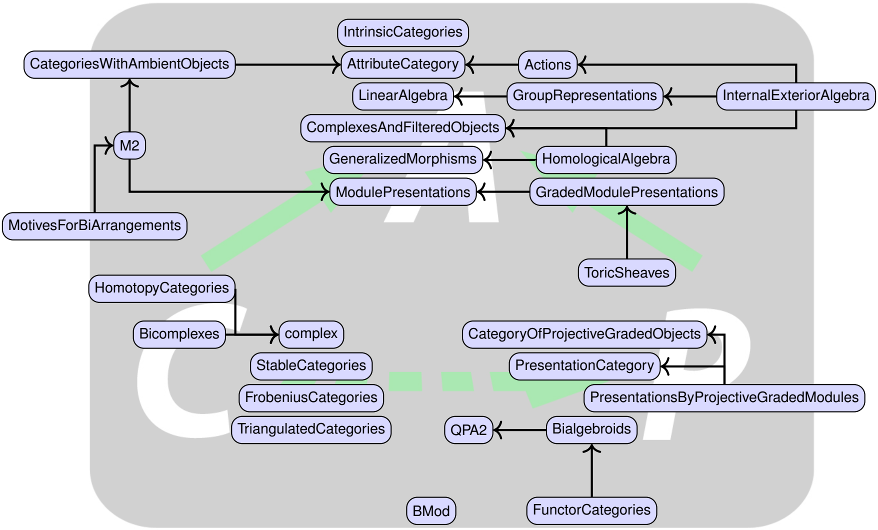
CAP packages



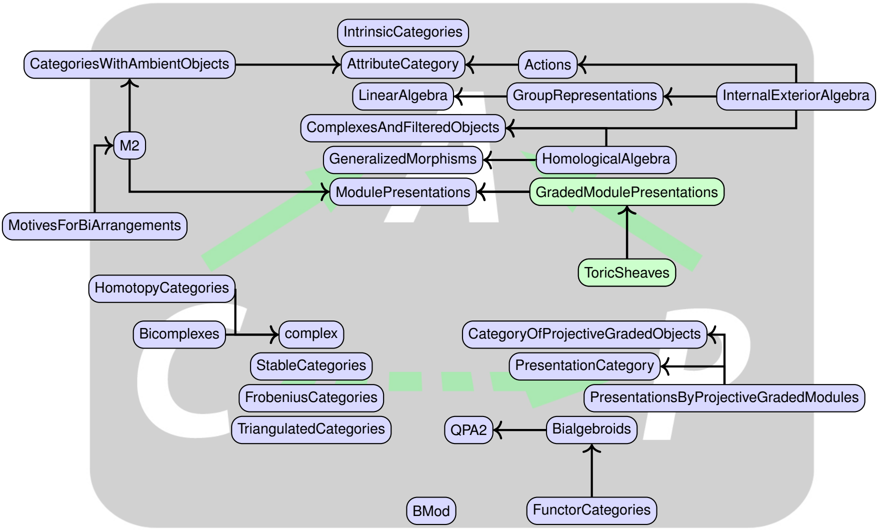
CAP packages



CAP packages



CAP packages



Applications to Algebraic Geometry

Coherent sheaves: Affine space

Let K be an algebraically closed field.

Coherent sheaves: Affine space

Let K be an algebraically closed field.

Affine space

- Affine space: $\mathbb{A}^n = K^n$

Coherent sheaves: Affine space

Let K be an algebraically closed field.

Affine space

- Affine space: $\mathbb{A}^n = K^n$
- Coherent sheaves correspond to f. g. modules over $S := K[x_1, \dots, x_n]$

Coherent sheaves: Affine space

Let K be an algebraically closed field.

Affine space

- Affine space: $\mathbb{A}^n = K^n$
- Coherent sheaves correspond to f. g. modules over $S := K[x_1, \ldots, x_n]$

In the language of category theory:

Coherent sheaves: Affine space

Let K be an algebraically closed field.

Affine space

- Affine space: $\mathbb{A}^n = K^n$
- Coherent sheaves correspond to f. g. modules over $S := K[x_1, \dots, x_n]$

In the language of category theory:
Equivalence of categories

$$S\text{-mod} \xrightarrow{\sim} \mathfrak{Coh}(\mathbb{A}^n)$$

Coherent sheaves

Coherent sheaves

Projective space

Coherent sheaves

Projective space

- Projective space $\mathbb{P}^{n-1} = (K^n/K^*) - \overline{\{0\}}$,

Coherent sheaves

Projective space

- Projective space $\mathbb{P}^{n-1} = (K^n / K^*) - \overline{\{0\}}$,

Coherent sheaves

Projective space

- Projective space $\mathbb{P}^{n-1} = (K^n/K^*) - \overline{\{0\}}$,
- Coherent sheaves correspond to f. g. modules over $S := K[x_1, \ldots, x_n]$

Coherent sheaves

Projective space

- Projective space $\mathbb{P}^{n-1} = (K^n/K^*) - \overline{\{0\}}$,
- Coherent sheaves correspond to f. g. modules over $S := K[x_1, \ldots, x_n]$

In the language of category theory:

Coherent sheaves

Projective space

- Projective space $\mathbb{P}^{n-1} = (K^n/K^*) - \overline{\{0\}}$,
- Coherent sheaves correspond to f. g. modules over $S := K[x_1, \ldots, x_n]$

In the language of category theory:

$$\mathfrak{Coh}\left(\mathbb{P}^{n-1}\right)$$

Coherent sheaves

Projective space

- Projective space $\mathbb{P}^{n-1} = (K^n / K^*) - \overline{\{0\}}$,
- Coherent sheaves correspond to f. g. modules over $S := K[x_1, \dots, x_n]$

In the language of category theory:

$S\text{-mod}$ $\qquad\qquad$ $\mathfrak{Coh}\left(\mathbb{P}^{n-1}\right)$

Coherent sheaves

Projective space

- Projective space $\mathbb{P}^{n-1} = (K^n / K^*) - \overline{\{0\}}$,
- Coherent sheaves correspond to f. g. modules over $S := K[x_1, \ldots, x_n]$

In the language of category theory:

S-mod $\qquad\qquad\qquad\qquad$ $\mathfrak{Coh}\left(\mathbb{P}^{n-1}\right)$

Coherent sheaves

Projective space

- Projective space $\mathbb{P}^{n-1} = (K^n / K^*) - \overline{\{0\}}$,
- Coherent sheaves correspond to f. g. modules over $S := K[x_1, \ldots, x_n]$ with a $\mathbb{Z}$-grading

In the language of category theory:

S-mod $\qquad\qquad$ $\mathfrak{Coh}\left(\mathbb{P}^{n-1}\right)$

Coherent sheaves

Projective space

- Projective space $\mathbb{P}^{n-1} = (K^n/K^*) - \overline{\{0\}}$, $K^* \cong \mathrm{Hom}\,(\mathbb{Z}, K^*)$
- Coherent sheaves correspond to f. g. modules over $S := K\,[x_1, \dots, x_n]$ with a $\mathbb{Z}$-grading

In the language of category theory:

S-mod $\qquad\qquad \mathfrak{Coh}\left(\mathbb{P}^{n-1}\right)$

Coherent sheaves

Projective space

- Projective space $\mathbb{P}^{n-1} = (K^n/K^*) - \overline{\{0\}}$, $K^* \cong \mathrm{Hom}\,(\mathbb{Z}, K^*)$
- Coherent sheaves correspond to f. g. modules over $S := K\,[x_1, \ldots, x_n]$ with a $\mathbb{Z}$-grading

In the language of category theory:

$S\text{-}\mathrm{grmod}_{\mathbb{Z}}$ $\qquad\qquad$ $\mathfrak{Coh}\left(\mathbb{P}^{n-1}\right)$

Coherent sheaves

Projective space

- Projective space $\mathbb{P}^{n-1} = (K^n / K^*) - \overline{\{0\}}$, $K^* \cong \mathrm{Hom}\,(\mathbb{Z}, K^*)$
- Coherent sheaves correspond to f. g. modules over $S := K\,[x_1, \ldots, x_n]$ with a $\mathbb{Z}$-grading

In the language of category theory:

$S\text{-}\mathrm{grmod}_{\mathbb{Z}}$ $\qquad\qquad$ $\mathfrak{Coh}\,\left(\mathbb{P}^{n-1}\right)$

Coherent sheaves

Projective space

- Projective space $\mathbb{P}^{n-1} = (K^n/K^*) - \overline{\{0\}}$, $K^* \cong \mathrm{Hom}(\mathbb{Z}, K^*)$
- Coherent sheaves correspond to f. g. modules over $S := K[x_1, \ldots, \underline{x_n}]$ with a $\mathbb{Z}$-grading modulo modules that are only supported on $\overline{\{0\}}$.

In the language of category theory:

$S\text{-grmod}_{\mathbb{Z}}$ $\qquad\qquad$ $\mathfrak{Coh}\left(\mathbb{P}^{n-1}\right)$

Coherent sheaves

Projective space

- Projective space $\mathbb{P}^{n-1} = (K^n / K^*) - \overline{\{0\}}$, $K^* \cong \mathrm{Hom}\left(\mathbb{Z}, K^*\right)$
- Coherent sheaves correspond to f. g. modules over $S := K\left[x_1, \ldots, x_n\right]$ with a $\mathbb{Z}$-grading modulo modules that are only supported on $\overline{\{0\}}$.

In the language of category theory:

$$S\text{-grmod}_{\mathbb{Z}} / S\text{-grmod}_{\mathbb{Z}}^{0} \qquad\qquad \mathfrak{Coh}\left(\mathbb{P}^{n-1}\right)$$

Coherent sheaves

Projective space

- Projective space $\mathbb{P}^{n-1} = (K^n/K^*) - \overline{\{0\}}$, $K^* \cong \mathrm{Hom}\,(\mathbb{Z}, K^*)$
- Coherent sheaves correspond to f. g. modules over $S := K\,[x_1, \ldots, x_n]$ with a $\mathbb{Z}$-grading modulo modules that are only supported on $\overline{\{0\}}$.

In the language of category theory:
Equivalence of categories

$$S\text{-grmod}_{\mathbb{Z}}/S\text{-grmod}_{\mathbb{Z}}^{0} \qquad\qquad \mathfrak{Coh}\left(\mathbb{P}^{n-1}\right)$$

Coherent sheaves

Projective space

- Projective space $\mathbb{P}^{n-1} = (K^n / K^*) - \overline{\{0\}}$, $K^* \cong \mathrm{Hom}(\mathbb{Z}, K^*)$
- Coherent sheaves correspond to f. g. modules over $S := K[x_1, \ldots, x_n]$ with a $\mathbb{Z}$-grading modulo modules that are only supported on $\overline{\{0\}}$.

In the language of category theory:
Equivalence of categories

$$S\text{-grmod}_{\mathbb{Z}} / S\text{-grmod}_{\mathbb{Z}}^{0} \xrightarrow{\sim} \mathfrak{Coh}\left(\mathbb{P}^{n-1}\right)$$

Coherent sheaves

Projective space

- Projective space $\mathbb{P}^{n-1} = (K^n/K^*) - \overline{\{0\}}$, $K^* \cong \mathrm{Hom}\,(\mathbb{Z}, K^*)$
- Coherent sheaves correspond to f. g. modules over $S := K\,[x_1, \ldots, x_n]$ with a $\mathbb{Z}$-grading modulo modules that are only supported on $\overline{\{0\}}$.

In the language of category theory:
Equivalence of categories

$$S\text{-}\mathrm{grmod}_{\mathbb{Z}} / S\text{-}\mathrm{grmod}^0_{\mathbb{Z}} \xrightarrow{\sim} \mathfrak{Coh}\left(\mathbb{P}^{n-1}\right)$$

Coherent sheaves

Normal toric variety (smooth)

- Projective space $\mathbb{P}^{n-1} = (K^n/K^*) - \overline{\{0\}}$, $K^* \cong \mathrm{Hom}\,(\mathbb{Z}, K^*)$
- Coherent sheaves correspond to f. g. modules over $S := K\,[x_1, \ldots, x_n]$ with a $\mathbb{Z}$-grading modulo modules that are only supported on $\overline{\{0\}}$.

In the language of category theory:
Equivalence of categories

$$S\text{-grmod}_{\mathbb{Z}} / S\text{-grmod}^0_{\mathbb{Z}} \xrightarrow{\sim} \mathfrak{Coh}\left(\mathbb{P}^{n-1}\right)$$

Coherent sheaves

Normal toric variety (smooth)

- Toric variety $X = (K^n/K^*) - \overline{\{0\}}$, $K^* \cong \mathrm{Hom}\left(\mathbb{Z}, K^*\right)$
- Coherent sheaves correspond to f. g. modules over $S := K\left[x_1, \ldots, x_n\right]$ with a $\mathbb{Z}$-grading modulo modules that are only supported on $\overline{\{0\}}$.

In the language of category theory:
Equivalence of categories

$$S\text{-grmod}_{\mathbb{Z}}/S\text{-grmod}_{\mathbb{Z}}^{0} \xrightarrow{\sim} \mathfrak{Coh}\left(\mathbb{P}^{n-1}\right)$$

Coherent sheaves

Normal toric variety (smooth)

- Toric variety $X = (K^n / G') - \overline{\{0\}}$, $G' \cong \mathrm{Hom}\,(G, K^*)$
- Coherent sheaves correspond to f. g. modules over $S := K\,[x_1, \ldots, x_n]$ with a $\mathbb{Z}$-grading modulo modules that are only supported on $\overline{\{0\}}$.

In the language of category theory:
Equivalence of categories

$$S\text{-grmod}_{\mathbb{Z}} / S\text{-grmod}^0_{\mathbb{Z}} \xrightarrow{\sim} \mathfrak{Coh}\left(\mathbb{P}^{n-1}\right)$$

Coherent sheaves

Normal toric variety (smooth)

- Toric variety $X = (K^n/G') - Z$, $G' \cong \mathrm{Hom}\,(G, K^*)$
- Coherent sheaves correspond to f. g. modules over $S := K\,[x_1, \ldots, x_n]$ with a $\mathbb{Z}$-grading modulo modules that are only supported on $\overline{\{0\}}$.

In the language of category theory:
Equivalence of categories

$$S\text{-grmod}_{\mathbb{Z}}/S\text{-grmod}^0_{\mathbb{Z}} \xrightarrow{\sim} \mathfrak{Coh}\left(\mathbb{P}^{n-1}\right)$$

Coherent sheaves

Normal toric variety (smooth)

- Toric variety $X = (K^n/G') - Z$, $G' \cong \mathrm{Hom}(G, K^*)$
- Coherent sheaves correspond to f. g. modules over $S := K[x_1, \dots, x_n]$ with a G-grading modulo modules that are only supported on $\overline{\{0\}}$.

In the language of category theory:
Equivalence of categories

$$S\text{-grmod}_{\mathbb{Z}} / S\text{-grmod}_{\mathbb{Z}}^0 \xrightarrow{\sim} \mathfrak{Coh}\left(\mathbb{P}^{n-1}\right)$$

Coherent sheaves

Normal toric variety (smooth)

- Toric variety $X = (K^n/G') - Z$, $G' \cong \mathrm{Hom}\left(G, K^*\right)$
- Coherent sheaves correspond to f. g. modules over $S := K\left[x_1, \ldots, x_n\right]$ with a G-grading modulo modules that are only supported on Z.

In the language of category theory:
Equivalence of categories

$$S\text{-grmod}_{\mathbb{Z}} / S\text{-grmod}_{\mathbb{Z}}^0 \xrightarrow{\sim} \mathfrak{Coh}\left(\mathbb{P}^{n-1}\right)$$

Coherent sheaves

Normal toric variety (smooth)

- Toric variety $X = (K^n/G') - Z$, $G' \cong \mathrm{Hom}\,(G, K^*)$
- Coherent sheaves correspond to f. g. modules over $S := K\,[x_1, \ldots, x_n]$ with a G-grading modulo modules that are only supported on Z.

In the language of category theory:
Equivalence of categories

$$S\text{-grmod}_{\mathbb{Z}} / S\text{-grmod}^0_{\mathbb{Z}} \xrightarrow{\sim} \mathfrak{Coh}\,(X)$$

Coherent sheaves

Normal toric variety (smooth)

- Toric variety $X = (K^n/G') - Z$, $G' \cong \mathrm{Hom}\,(G, K^*)$
- Coherent sheaves correspond to f. g. modules over $S := K\,[x_1, \ldots, x_n]$ with a G-grading modulo modules that are only supported on Z.

In the language of category theory:
Equivalence of categories

$$S\text{-grmod}_G / S\text{-grmod}_G^0 \xrightarrow{\sim} \mathfrak{Coh}\,(X)$$

Coherent sheaves

Normal toric variety (smooth)

- Toric variety $X = (K^n/G') - Z$, $G' \cong \mathrm{Hom}\,(G, K^*)$
- Coherent sheaves correspond to f. g. modules over $S := K\,[x_1, \ldots, x_n]$ with a G-grading modulo modules that are only supported on Z.

In the language of category theory:
Equivalence of categories

$$S\text{-grmod}_G / S\text{-grmod}_G^0 \xrightarrow{\sim} \mathfrak{Coh}\,(X)$$

Coherent sheaves

Normal toric variety

- Toric variety $X = (K^n/G') - Z$, $G' \cong \mathrm{Hom}\,(G, K^*)$
- Coherent sheaves correspond to f. g. modules over $S := K[x_1, \dots, x_n]$ with a G-grading modulo modules that are only supported on Z.

In the language of category theory:
Equivalence of categories

$$S\text{-grmod}_G / S\text{-grmod}_G^0 \xrightarrow{\sim} \mathfrak{Coh}\,(X)$$

Coherent sheaves

Normal toric variety

- Toric variety $X = (K^n/G') - Z$, $G' \cong \mathrm{Hom}\,(G, K^*)$
- Coherent sheaves correspond to f. g. modules over $S := K\,[x_1, \ldots, x_n]$ with a G-grading modulo modules that sheafify to zero.

In the language of category theory:
Equivalence of categories

$$S\text{-grmod}_G / S\text{-grmod}_G^0 \xrightarrow{\sim} \mathfrak{Coh}\,(X)$$

Coherent sheaves

Normal toric variety

- Toric variety $X = (K^n/G') - Z$, $G' \cong \mathrm{Hom}(G, K^*)$
- Coherent sheaves correspond to f. g. modules over $S := K[x_1, \ldots, x_n]$ with a G-grading modulo modules that sheafify to zero.

In the language of category theory:
Equivalence of categories

$$S\text{-grmod}_G / S\text{-grmod}_G^0 \xrightarrow{\sim} \mathfrak{Coh}(X)$$

Computability of $S\text{-grmod}_G / S\text{-grmod}_G^0$?

Serre quotients

Serre quotient

Serre quotients

Serre quotient

Let $\mathcal{A}$ be an abelian category and $\mathcal{C}$ a thick subcategory.

Serre quotients

Serre quotient

Let $\mathcal{A}$ be an abelian category and $\mathcal{C}$ a thick subcategory. The **Serre quotient** $\mathcal{A}/\mathcal{C}$ is an abelian category with

Serre quotients

Serre quotient

Let $\mathcal{A}$ be an abelian category and $\mathcal{C}$ a thick subcategory. The **Serre quotient** $\mathcal{A}/\mathcal{C}$ is an abelian category with

- $\mathrm{Obj}_{\mathcal{A}/\mathcal{C}} := \mathrm{Obj}_{\mathcal{A}}$

Serre quotients

Serre quotient

Let $\mathcal{A}$ be an abelian category and $\mathcal{C}$ a thick subcategory. The **Serre quotient** $\mathcal{A}/\mathcal{C}$ is an abelian category with

- $\mathrm{Obj}_{\mathcal{A}/\mathcal{C}} := \mathrm{Obj}_{\mathcal{A}}$
- $\mathrm{Hom}_{\mathcal{A}/\mathcal{C}}(A, B) :=$

$$\left\{ \begin{tikzcd} A \arrow[rr, dashed] & & B \\ & X \arrow[ul, "\psi"] \arrow[ur, "\varphi"'] & \end{tikzcd} \;\middle|\; \qquad \right\}$$

Serre quotients

Serre quotient

Let $\mathcal{A}$ be an abelian category and $\mathcal{C}$ a thick subcategory. The **Serre quotient** $\mathcal{A}/\mathcal{C}$ is an abelian category with

- $\mathrm{Obj}_{\mathcal{A}/\mathcal{C}} := \mathrm{Obj}_{\mathcal{A}}$
- $\mathrm{Hom}_{\mathcal{A}/\mathcal{C}}(A, B) :=$

$$\left\{ \begin{tikzcd} A \arrow[rr, dashed] & & B \\ & X \arrow[ul, "\psi"] \arrow[ur, "\varphi"'] & \end{tikzcd} \;\middle|\; \mathrm{coker}(\psi) \in \mathcal{C} \right\}$$

Serre quotients

Serre quotient

Let $\mathcal{A}$ be an abelian category and $\mathcal{C}$ a thick subcategory. The **Serre quotient** $\mathcal{A}/\mathcal{C}$ is an abelian category with

- $\mathrm{Obj}_{\mathcal{A}/\mathcal{C}} := \mathrm{Obj}_{\mathcal{A}}$
- $\mathrm{Hom}_{\mathcal{A}/\mathcal{C}}(A, B) :=$

$$\left\{ \begin{tikzcd} A \arrow[rr, dashed] & & B \\ & X \arrow[ul, "\psi"] \arrow[ur, "\varphi"'] & \end{tikzcd} \;\middle|\; \begin{array}{l} \mathrm{coker}(\psi) \in \mathcal{C} \\ \varphi(\ker(\psi)) \in \mathcal{C} \end{array} \right\}$$

Serre quotients

Serre quotient

Let $\mathcal{A}$ be an abelian category and $\mathcal{C}$ a thick subcategory. The **Serre quotient** $\mathcal{A}/\mathcal{C}$ is an abelian category with

- $\mathrm{Obj}_{\mathcal{A}/\mathcal{C}} := \mathrm{Obj}_{\mathcal{A}}$
- $\mathrm{Hom}_{\mathcal{A}/\mathcal{C}}(A, B) :=$

$$\left. \left\{ \begin{tikzcd} A \arrow[rr, dashed] & & B \\ & X \arrow[ul, "\psi"] \arrow[ur, "\varphi"'] & \end{tikzcd} \;\middle|\; \begin{array}{l} \mathrm{coker}(\psi) \in \mathcal{C} \\ \varphi(\ker(\psi)) \in \mathcal{C} \end{array} \right\} \middle/ \sim \right.$$

Composition in the Serre quotient

Composition in the Serre quotient

Composition in the Serre quotient $\mathcal{A}/\mathcal{C}$

Composition in the Serre quotient

Composition in the Serre quotient $\mathcal{A}/\mathcal{C}$

$$\begin{tikzcd}
A \arrow[rr, dashed] & & B \arrow[rr, dashed] & & C \\
& X \arrow[ul] \arrow[ur] & & Y \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

Composition in the Serre quotient

Composition in the Serre quotient $\mathcal{A}/\mathcal{C}$

$$
\begin{tikzcd}
A \arrow[rr, dashed] & & B \arrow[rr, dashed] & & C \\
& X \arrow[ul] \arrow[ur] & & Y \arrow[ul] \arrow[ur] &
\end{tikzcd}
$$

Composition in the Serre quotient

Composition in the Serre quotient $\mathcal{A}/\mathcal{C}$

$$
\begin{tikzcd}
A \arrow[rr, dashed] & & {\color{red}B} \arrow[rr, dashed] & & C \\
& {\color{red}X} \arrow[ul] \arrow[ur, red] & & {\color{red}Y} \arrow[ul, red] \arrow[ur] &
\end{tikzcd}
$$

Composition in the Serre quotient

Composition in the Serre quotient $\mathcal{A}/\mathcal{C}$

$$
\begin{tikzcd}
A \arrow[rr, dashed] & & B \arrow[rr, dashed] & & C \\
& X \arrow[ul] \arrow[ur] & & Y \arrow[ul] \arrow[ur] & \\
& & Z \arrow[ul] \arrow[ur] & &
\end{tikzcd}
$$

Composition in the Serre quotient

Composition in the Serre quotient $\mathcal{A}/\mathcal{C}$

$$
\begin{tikzcd}
A \arrow[rr, dashed] & & B \arrow[rr, dashed] & & C \\
& X \arrow[ul] \arrow[ur] & & Y \arrow[ul] \arrow[ur] & \\
& & Z \arrow[ul] \arrow[ur] & &
\end{tikzcd}
$$

`FiberProduct`: Algorithm for intersection

Composition in the Serre quotient

Composition in the Serre quotient $\mathcal{A}/\mathcal{C}$

$$
\begin{tikzcd}
A \arrow[rr, dashed] & & B \arrow[rr, dashed] & & C \\
& X \arrow[ul] \arrow[ur] & & Y \arrow[ul] \arrow[ur] & \\
& & Z \arrow[ul] \arrow[ur] & &
\end{tikzcd}
$$

Composition in the Serre quotient

Composition in the Serre quotient $\mathcal{A}/\mathcal{C}$

$$
\begin{tikzcd}
A \arrow[rr, dashed] & & B \arrow[rr, dashed] & & C \\
& X \arrow[ul] \arrow[ur] & & Y \arrow[ul] \arrow[ur] & \\
& & Z \arrow[ul] \arrow[ur] & &
\end{tikzcd}
$$

Composition only by computations in $\mathcal{A}$!

Computability of toric coherent sheaves

Computability of toric coherent sheaves

Theorem (Barakat, Lange-Hegermann)

Is $\mathcal{A}$ *computable* abelian and $\mathcal{C}$ *decidable*,

Computability of toric coherent sheaves

Theorem (Barakat, Lange-Hegermann)

Is $\mathcal{A}$ *computable* abelian and $\mathcal{C}$ *decidable*, then $\mathcal{A}/\mathcal{C}$ is computable abelian.

Computability of toric coherent sheaves

Theorem (Barakat, Lange-Hegermann)

Is $\mathcal{A}$ *computable* abelian and $\mathcal{C}$ *decidable*, then $\mathcal{A}/\mathcal{C}$ is computable abelian.

$$S\text{-grmod}_G / S\text{-grmod}_G^0 \cong \mathfrak{Coh}\,(X)\ ?$$

Computability of toric coherent sheaves

Theorem (Barakat, Lange-Hegermann)

Is $\mathcal{A}$ *computable* abelian and $\mathcal{C}$ *decidable*, then $\mathcal{A}/\mathcal{C}$ is computable abelian.

$$S\text{-grmod}_G / S\text{-grmod}_G^0 \cong \mathfrak{Coh}\,(X) \text{ ?}$$

Theorem (G.)

Let X be a normal toric variety without torus factors.

Computability of toric coherent sheaves

Theorem (Barakat, Lange-Hegermann)

Is $\mathcal{A}$ *computable* abelian and $\mathcal{C}$ *decidable*, then $\mathcal{A}/\mathcal{C}$ is computable abelian.

$$S\text{-}\mathrm{grmod}_G / S\text{-}\mathrm{grmod}_G^0 \cong \mathfrak{Coh}\,(X)\ ?$$

Theorem (G.)

Let X be a normal toric variety without torus factors. Then the thick subcategory $S\text{-}\mathrm{grmod}_G^0$ of f. g. G-graded modules over S which sheafify to zero

Computability of toric coherent sheaves

Theorem (Barakat, Lange-Hegermann)

Is $\mathcal{A}$ *computable* abelian and $\mathcal{C}$ *decidable*, then $\mathcal{A}/\mathcal{C}$ is computable abelian.

$$S\text{-}\mathrm{grmod}_G / S\text{-}\mathrm{grmod}_G^0 \cong \mathfrak{Coh}\,(X)\ ?$$

Theorem (G.)

Let X be a normal toric variety without torus factors. Then the thick subcategory $S\text{-}\mathrm{grmod}_G^0$ of f. g. G-graded modules over S which sheafify to zero is decidable.

Computability of toric coherent sheaves

Theorem (Barakat, Lange-Hegermann)

Is $\mathcal{A}$ *computable* abelian and $\mathcal{C}$ *decidable*, then $\mathcal{A}/\mathcal{C}$ is computable abelian.

$$S\text{-}\mathrm{grmod}_G / S\text{-}\mathrm{grmod}_G^0 \cong \mathfrak{Coh}\,(X) \text{ ?}$$

Theorem (G.)

Let X be a normal toric variety without torus factors. Then the thick subcategory $S\text{-}\mathrm{grmod}_G^0$ of f. g. G-graded modules over S which sheafify to zero is decidable.

So $\mathfrak{Coh}\,(X)$ is computable abelian!

Coherent sheaves

So $\mathfrak{Coh}\,(X)$ is computable abelian!

Coherent sheaves

So $\mathfrak{Coh}\,(X)$ is computable abelian!

We can apply algorithms for abelian categories to coherent sheaves over toric varieties:

Coherent sheaves

So $\mathfrak{Coh}\,(X)$ is computable abelian!

We can apply algorithms for abelian categories to coherent sheaves over toric varieties:

- Intersection

Coherent sheaves

So $\mathfrak{Coh}\,(X)$ is computable abelian!

We can apply algorithms for abelian categories to coherent sheaves over toric varieties:

- Intersection
- Homology

Coherent sheaves

So $\mathfrak{Coh}\,(X)$ is computable abelian!

We can apply algorithms for abelian categories to coherent sheaves over toric varieties:

- Intersection
- Homology
- Diagram chases

Coherent sheaves

So $\mathfrak{Coh}\,(X)$ is computable abelian!

We can apply algorithms for abelian categories to coherent sheaves over toric varieties:

- Intersection
- Homology
- Diagram chases
- Spectral sequences

Coherent sheaves

So $\mathfrak{Coh}\,(X)$ is computable abelian!

We can apply algorithms for abelian categories to coherent sheaves over toric varieties:

- Intersection
- Homology
- Diagram chases
- Spectral sequences
- Purity filtration

Coherent sheaves

So $\mathfrak{Coh}\,(X)$ is computable abelian!

We can apply algorithms for abelian categories to coherent sheaves over toric varieties:

- Intersection
- Homology
- Diagram chases
- Spectral sequences
- Purity filtration
- ...