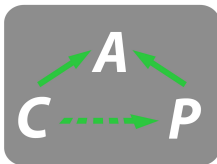


A constructive model for coherent sheaves over a normal toric variety

Sebastian Gutsche
(j/w Sebastian Posur)

University of Siegen

Paderborn, March 6, 2018



1 Computable Categories

- 1 Computable Categories
- 2 Model for coherent sheaves over normal toric varieties

Section 1

Computable Categories

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

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)

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

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)

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

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

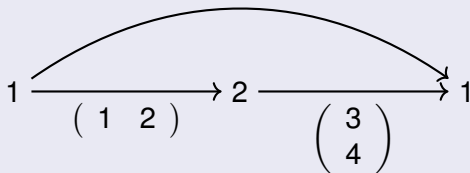
Computable categories

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)

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



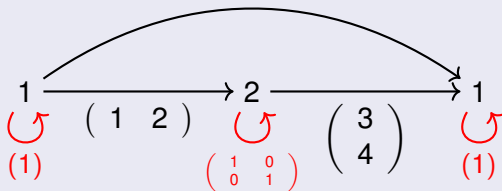
Computable categories

A category becomes computable through

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

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

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



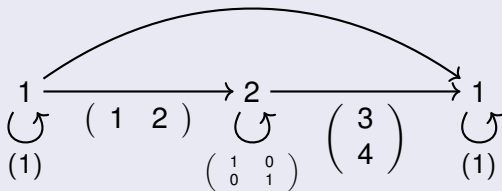
Computable categories

A category becomes computable through

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

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

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



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

Implementation of the kernel

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

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

Implementation of the kernel

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

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

Implementation of the kernel

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

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

$\ker \varphi$

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

Implementation of the kernel

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

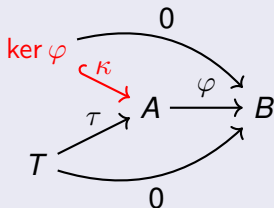
\dots one needs an object $\text{ker } \varphi$,
its embedding $\kappa = \text{KernelEmbedding}(\varphi)$,

$$\text{ker } \varphi \xrightarrow{\kappa} A \xrightarrow{\varphi} B$$

Implementation of the kernel

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

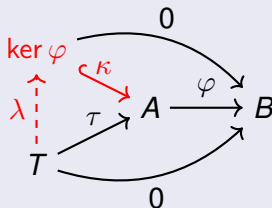
\dots one needs an object $\text{ker } \varphi$,
 its embedding $\kappa = \text{KernelEmbedding}(\varphi)$,
 and for every test morphism τ



Implementation of the kernel

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

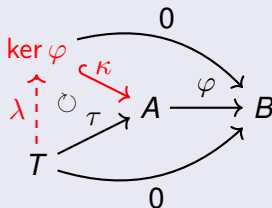
\dots one needs an object $\text{ker } \varphi$,
 its embedding $\kappa = \text{KernelEmbedding}(\varphi)$,
 and for every test morphism τ
 a *unique* morphism $\lambda = \text{KernelLift}(\varphi, \tau)$



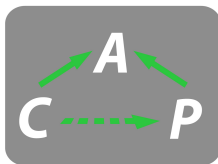
Implementation of the kernel

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

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

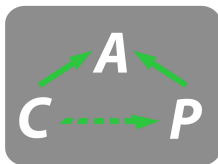


The CAP - project



CAP - Categories, Algorithms, Programming

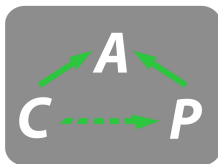
The CAP - project



CAP - Categories, Algorithms, Programming

CAP is a framework written in GAP

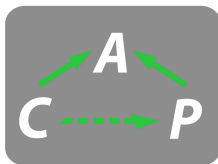
The CAP - project



CAP - Categories, Algorithms, Programming

CAP is a framework written in GAP to implement computable categories and provides

The CAP - project

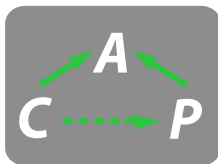


CAP - Categories, Algorithms, Programming

CAP is a framework written in GAP to implement computable categories and provides

- specifications of categorical operations,

The CAP - project

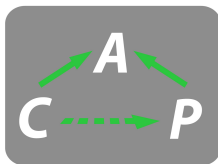


CAP - Categories, Algorithms, Programming

CAP is a framework written in GAP to implement computable categories and provides

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

The CAP - project

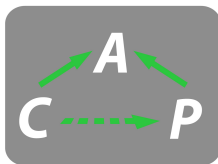


CAP - Categories, Algorithms, Programming

CAP is a framework written in GAP 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.

The CAP - project



CAP - Categories, Algorithms, Programming

CAP is a framework written in GAP 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

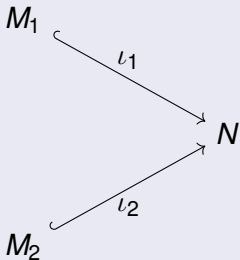
Let $M_1 \hookrightarrow N$ and $M_2 \hookrightarrow N$ subobjects.

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.

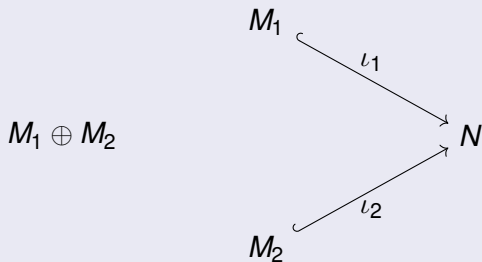
Compute their intersection $\gamma : M_1 \cap M_2 \hookrightarrow N$.



Computing the intersection

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

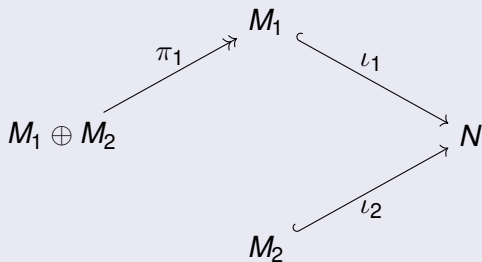
Compute their intersection $\gamma : M_1 \cap M_2 \hookrightarrow N$.



Computing the intersection

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

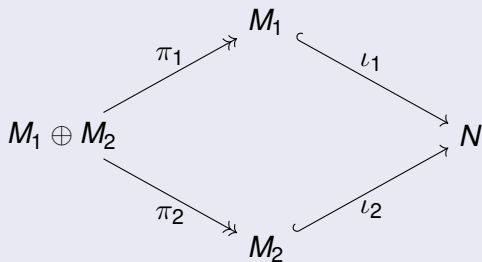
Compute their intersection $\gamma : M_1 \cap M_2 \hookrightarrow N$.



Computing the intersection

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

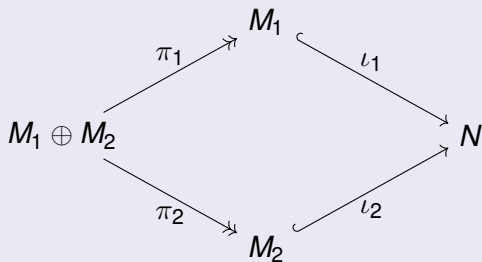
Compute their intersection $\gamma : M_1 \cap M_2 \hookrightarrow N$.



Computing the intersection

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

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

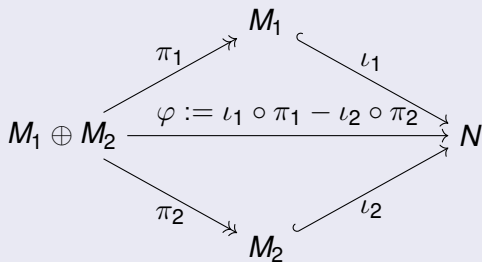


- $\pi_i := \text{ProjectionInFactorOfDirectSum}((M_1, M_2), i), i = 1, 2$

Computing the intersection

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

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

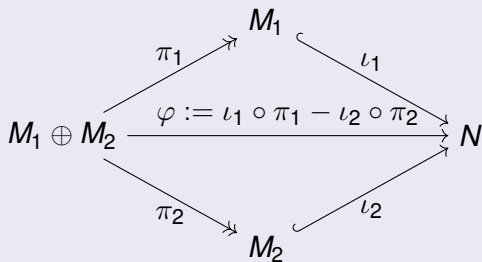


- $\pi_i := \text{ProjectionInFactorOfDirectSum}((M_1, M_2), i), i = 1, 2$

Computing the intersection

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

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

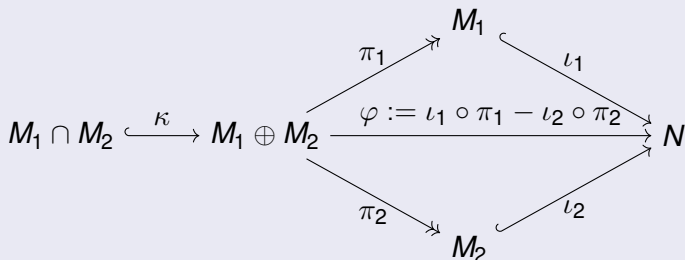


- $\pi_i := \text{ProjectionInFactorOfDirectSum}((M_1, M_2), i), i = 1, 2$
- $\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$

Computing the intersection

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

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



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

Computing the intersection

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

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

$$\begin{array}{ccccc}
 & & & M_1 & \\
 & & \nearrow \pi_1 & & \searrow \iota_1 \\
 M_1 \cap M_2 & \xrightarrow{\kappa} & M_1 \oplus M_2 & & N \\
 & & \searrow \pi_2 & & \nearrow \iota_2 \\
 & & & M_2 &
 \end{array}$$

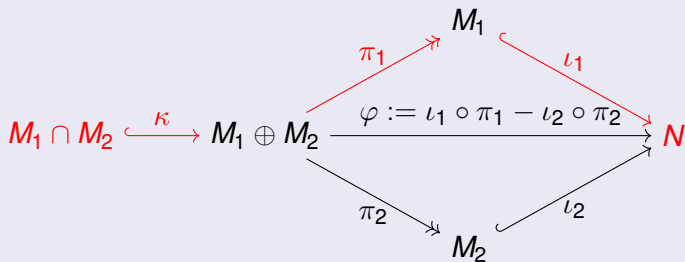
$\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$

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

Computing the intersection

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

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

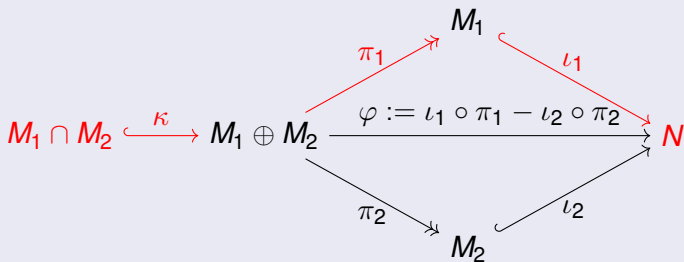


- $\pi_i := \text{ProjectionInFactorOfDirectSum}((M_1, M_2), i), i = 1, 2$
- $\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$
- $\kappa := \text{KernelEmbedding}(\varphi)$

Computing the intersection

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

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



- $\pi_i := \text{ProjectionInFactorOfDirectSum}((M_1, M_2), i), i = 1, 2$
- $\varphi := l_1 \circ \pi_1 - l_2 \circ \pi_2$
- $\kappa := \text{KernelEmbedding}(\varphi)$
- $\gamma := l_1 \circ \pi_1 \circ \kappa$

Translation to CAP

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

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

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

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

Translation to CAP

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

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

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

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

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

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

Translation to CAP

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

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

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

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

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

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

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

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

Translation to CAP

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

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

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

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

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

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

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

`kappa := KernelEmbedding(phi);`

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

Translation to CAP

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

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

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

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

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

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

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

`kappa := KernelEmbedding(phi);`

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

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

Translation to CAP

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

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

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

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

Translation to CAP

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

Translation to CAP

```
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;
```

Section 2

Model for coherent sheaves over normal toric varieties

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 $S := K[x_1, \dots, x_n]$

Coherent sheaves

Projective space

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

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 $S := K[x_1, \dots, x_n]$

In the language of category theory:

$$\mathcal{Coh}(\mathbb{P}^{n-1})$$

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:

$S\text{-mod}$

$\mathcal{Coh}(\mathbb{P}^{n-1})$

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:

$S\text{-mod}$

$\mathcal{Coh}(\mathbb{P}^{n-1})$

Coherent sheaves

Projective space

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

In the language of category theory:

$S\text{-mod}$

$\mathcal{Coh}(\mathbb{P}^{n-1})$

Coherent sheaves

Projective space

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

$S\text{-mod}$

$\mathcal{Coh}(\mathbb{P}^{n-1})$

Coherent sheaves

Projective space

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

$$\mathcal{S}\text{-grmod}_{\mathbb{Z}}$$

$$\mathcal{Coh}(\mathbb{P}^{n-1})$$

Coherent sheaves

Projective space

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

$$\mathcal{S}\text{-grmod}_{\mathbb{Z}}$$

$$\mathcal{Coh}(\mathbb{P}^{n-1})$$

Coherent sheaves

Projective space

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

In the language of category theory:

$$\mathcal{S}\text{-grmod}_{\mathbb{Z}}$$

$$\mathcal{Coh}(\mathbb{P}^{n-1})$$

Coherent sheaves

Projective space

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

In the language of category theory:

$$\mathcal{S}\text{-grmod}_{\mathbb{Z}} / \mathcal{S}\text{-grmod}_{\mathbb{Z}}^0$$

$$\mathcal{Coh}(\mathbb{P}^{n-1})$$

Coherent sheaves

Projective space

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

In the language of category theory:

Equivalence of categories

$$\mathcal{S}\text{-grmod}_{\mathbb{Z}} / \mathcal{S}\text{-grmod}_{\mathbb{Z}}^0$$

$$\mathcal{Coh}(\mathbb{P}^{n-1})$$

Coherent sheaves

Projective space

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

In the language of category theory:

Equivalence of categories

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

Coherent sheaves

Projective space

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

In the language of category theory:
Equivalence of categories

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

Coherent sheaves

Normal toric variety (smooth)

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

In the language of category theory:
Equivalence of categories

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

Coherent sheaves

Normal toric variety (smooth)

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

In the language of category theory:
Equivalence of categories

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

Coherent sheaves

Normal toric variety (smooth)

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

In the language of category theory:
Equivalence of categories

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

Coherent sheaves

Normal toric variety (smooth)

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

In the language of category theory:
Equivalence of categories

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

Coherent sheaves

Normal toric variety (smooth)

- Toric variety $X = (K^n/G') - Z$, $G' \cong \text{Hom}(G, K^*)$
- 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 $\{0\}$.

In the language of category theory:
Equivalence of categories

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

Coherent sheaves

Normal toric variety (smooth)

- Toric variety $X = (K^n/G') - Z$, $G' \cong \text{Hom}(G, K^*)$
- 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

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

Coherent sheaves

Normal toric variety (smooth)

- Toric variety $X = (K^n/G') - Z$, $G' \cong \text{Hom}(G, K^*)$
- 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

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

Coherent sheaves

Normal toric variety (smooth)

- Toric variety $X = (K^n/G') - Z$, $G' \cong \text{Hom}(G, K^*)$
- 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

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

Coherent sheaves

Normal toric variety (smooth)

- Toric variety $X = (K^n/G') - Z$, $G' \cong \text{Hom}(G, K^*)$
- 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

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

Coherent sheaves

Normal toric variety

- Toric variety $X = (K^n/G') - Z$, $G' \cong \text{Hom}(G, K^*)$
- 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

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

Coherent sheaves

Normal toric variety

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

In the language of category theory:
Equivalence of categories

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

Coherent sheaves

Normal toric variety

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

In the language of category theory:
Equivalence of categories

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

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

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

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

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

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

$$\left\{ \begin{array}{ccc} A & \overset{\text{---}}{\longrightarrow} & B \\ \swarrow \psi & & \nearrow \varphi \\ & X & \end{array} \right\} \quad \Bigg| \quad \left. \right\}$$

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

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

$$\left\{ \begin{array}{ccc} A & \overset{\text{---}}{\longrightarrow} & B \\ & \swarrow \psi & \nearrow \varphi \\ & X & \end{array} \right\} \quad \left| \quad \text{coker}(\psi) \in \mathcal{C} \right.$$

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

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

$$\left\{ \begin{array}{ccc} A & \overset{\text{---}}{\longrightarrow} & B \\ & \swarrow \psi & \nearrow \varphi \\ & X & \end{array} \right\} \left| \begin{array}{l} \text{coker}(\psi) \in \mathcal{C} \\ \varphi(\ker(\psi)) \in \mathcal{C} \end{array} \right\}$$

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

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

$$\left\{ \begin{array}{ccc} A & \overset{\text{---}}{\longrightarrow} & B \\ & \swarrow \psi & \nearrow \varphi \\ & X & \end{array} \right\} \left| \begin{array}{l} \text{coker}(\psi) \in \mathcal{C} \\ \varphi(\ker(\psi)) \in \mathcal{C} \end{array} \right\} / \sim$$

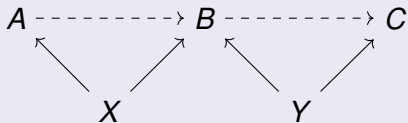
Composition in the Serre quotient

Composition in the Serre quotient

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

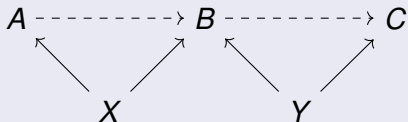
Composition in the Serre quotient

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



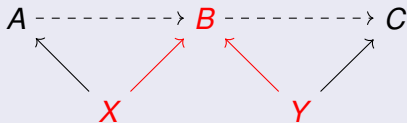
Composition in the Serre quotient

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



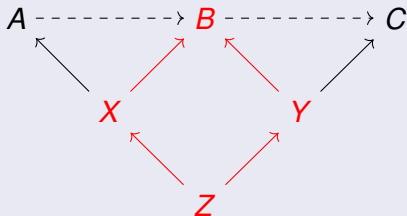
Composition in the Serre quotient

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



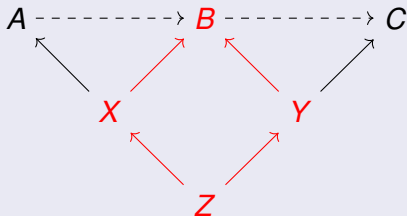
Composition in the Serre quotient

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



Composition in the Serre quotient

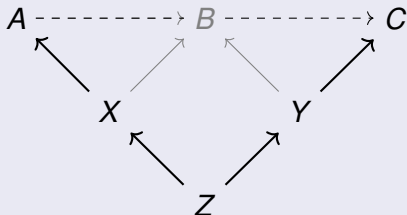
Composition in the Serre quotient \mathcal{A}/\mathcal{C}



FiberProduct: Algorithm for intersection

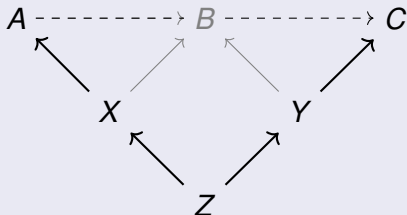
Composition in the Serre quotient

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



Composition in the Serre quotient

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



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.

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

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.

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

Decidability of $S\text{-grmod}_G^0$

Every toric variety X with Cox ring S has a finite affine cover $\{U_\tau\}$, defined by the orbits of the torus acting on X , and naturally indexed by monomials $\tau \in \text{Mon}(S)$.

Decidability of $S\text{-grmod}_G^0$

Every toric variety X with Cox ring S has a finite affine cover $\{U_\tau\}$, defined by the orbits of the torus acting on X , and naturally indexed by monomials $\tau \in \text{Mon}(S)$.

Let $M \in S\text{-grmod}_G$.

Decidability of $\mathcal{S}\text{-grmod}_G^0$

Every toric variety X with Cox ring \mathcal{S} has a finite affine cover $\{U_\tau\}$, defined by the orbits of the torus acting on X , and naturally indexed by monomials $\tau \in \text{Mon}(\mathcal{S})$.

Let $M \in \mathcal{S}\text{-grmod}_G$. We have $M \in \mathcal{S}\text{-grmod}_G^0$ iff

$$\Gamma(\tilde{M}, U_\tau)$$

Decidability of $\mathcal{S}\text{-grmod}_G^0$

Every toric variety X with Cox ring \mathcal{S} has a finite affine cover $\{U_\tau\}$, defined by the orbits of the torus acting on X , and naturally indexed by monomials $\tau \in \text{Mon}(\mathcal{S})$.

Let $M \in \mathcal{S}\text{-grmod}_G$. We have $M \in \mathcal{S}\text{-grmod}_G^0$ iff

$$\Gamma(\tilde{M}, U_\tau) = (M_\tau)_0$$

Decidability of $\mathcal{S}\text{-grmod}_G^0$

Every toric variety X with Cox ring \mathcal{S} has a finite affine cover $\{U_\tau\}$, defined by the orbits of the torus acting on X , and naturally indexed by monomials $\tau \in \text{Mon}(\mathcal{S})$.

Let $M \in \mathcal{S}\text{-grmod}_G$. We have $M \in \mathcal{S}\text{-grmod}_G^0$ iff

$$\Gamma(\tilde{M}, U_\tau) = (M_\tau)_0 = 0 \text{ for all } U_\tau.$$

Decidability of $\mathcal{S}\text{-grmod}_G^0$

Every toric variety X with Cox ring S has a finite affine cover $\{U_\tau\}$, defined by the orbits of the torus acting on X , and naturally indexed by monomials $\tau \in \text{Mon}(S)$.

Let $M \in \mathcal{S}\text{-grmod}_G$. We have $M \in \mathcal{S}\text{-grmod}_G^0$ iff

$$\Gamma(\tilde{M}, U_\tau) = (M_\tau)_0 = 0 \text{ for all } U_\tau.$$

Given a presentation of M , how to compute a presentation of $(M_\tau)_0$ for a given τ ?

Presentation of $(M_\tau)_0$

Strategy for computing a presentation of $(M_\tau)_0$ from a free graded presentation of M :

$$\prod_{i \in I} \mathcal{S}(\alpha_i) \xrightarrow{\varphi} \prod_{j \in J} \mathcal{S}(\beta_j) \longrightarrow M \longrightarrow 0.$$

Presentation of $(M_\tau)_0$

Strategy for computing a presentation of $(M_\tau)_0$ from a free graded presentation of M :

$$\prod_{i \in I} \mathcal{S}(\alpha_i) \xrightarrow{\varphi} \prod_{j \in J} \mathcal{S}(\beta_j) \longrightarrow M \longrightarrow 0.$$

Algorithm

Presentation of $(M_\tau)_0$

Strategy for computing a presentation of $(M_\tau)_0$ from a free graded presentation of M :

$$\prod_{i \in I} \mathcal{S}(\alpha_i) \xrightarrow{\varphi} \prod_{j \in J} \mathcal{S}(\beta_j) \longrightarrow M \longrightarrow 0.$$

Algorithm

- 1 Compute algebra generators for $(\mathcal{S}_\tau)_0$

Presentation of $(M_\tau)_0$

Strategy for computing a presentation of $(M_\tau)_0$ from a free graded presentation of M :

$$\prod_{i \in I} \mathcal{S}(\alpha_i) \xrightarrow{\varphi} \prod_{j \in J} \mathcal{S}(\beta_j) \longrightarrow M \longrightarrow 0.$$

Algorithm

- 1 Compute algebra generators for $(\mathcal{S}_\tau)_0$
 - 1 Compute inequalities for the cone $\text{Mon}((\mathcal{S}_\tau)_0)$.

Presentation of $(M_\tau)_0$

Strategy for computing a presentation of $(M_\tau)_0$ from a free graded presentation of M :

$$\prod_{i \in I} \mathcal{S}(\alpha_i) \xrightarrow{\varphi} \prod_{j \in J} \mathcal{S}(\beta_j) \longrightarrow M \longrightarrow 0.$$

Algorithm

- 1 Compute algebra generators for $(\mathcal{S}_\tau)_0$
 - 1 Compute inequalities for the cone $\text{Mon}((\mathcal{S}_\tau)_0)$.
 - 2 Compute the generators as HILBERT basis of $\text{Mon}((\mathcal{S}_\tau)_0)$.

Presentation of $(M_\tau)_0$

Strategy for computing a presentation of $(M_\tau)_0$ from a free graded presentation of M :

$$\prod_{i \in I} \mathcal{S}(\alpha_i) \xrightarrow{\varphi} \prod_{j \in J} \mathcal{S}(\beta_j) \longrightarrow M \longrightarrow 0.$$

Algorithm

- 1 Compute algebra generators for $(\mathcal{S}_\tau)_0$

Presentation of $(M_\tau)_0$

Strategy for computing a presentation of $(M_\tau)_0$ from a free graded presentation of M :

$$\prod_{i \in I} \mathcal{S}(\alpha_i) \xrightarrow{\varphi} \prod_{j \in J} \mathcal{S}(\beta_j) \longrightarrow M \longrightarrow 0.$$

Algorithm

- 1 Compute algebra generators for $(\mathcal{S}_\tau)_0$
- 2 Compute $(\mathcal{S}_\tau)_0$ -generators of $(\mathcal{S}(\gamma)_\tau)_0$, $\gamma \in \{\alpha_i, \beta_j\}$

Presentation of $(M_\tau)_0$

Strategy for computing a presentation of $(M_\tau)_0$ from a free graded presentation of M :

$$\prod_{i \in I} \mathcal{S}(\alpha_i) \xrightarrow{\varphi} \prod_{j \in J} \mathcal{S}(\beta_j) \longrightarrow M \longrightarrow 0.$$

Algorithm

- 1 Compute algebra generators for $(\mathcal{S}_\tau)_0$
- 2 Compute $(\mathcal{S}_\tau)_0$ -generators of $(\mathcal{S}(\gamma)_\tau)_0$, $\gamma \in \{\alpha_i, \beta_j\}$

$$\text{Mon}((\mathcal{S}(\gamma)_\tau)_0) = \underbrace{P'}_{\text{finite}} + \text{Mon}((\mathcal{S}_\tau)_0)$$

Presentation of $(M_\tau)_0$

Strategy for computing a presentation of $(M_\tau)_0$ from a free graded presentation of M :

$$\prod_{i \in I} \mathcal{S}(\alpha_i) \xrightarrow{\varphi} \prod_{j \in J} \mathcal{S}(\beta_j) \longrightarrow M \longrightarrow 0.$$

Algorithm

- 1 Compute algebra generators for $(\mathcal{S}_\tau)_0$
- 2 Compute $(\mathcal{S}_\tau)_0$ -generators of $(\mathcal{S}(\gamma)_\tau)_0$, $\gamma \in \{\alpha_i, \beta_j\}$

Presentation of $(M_\tau)_0$

Strategy for computing a presentation of $(M_\tau)_0$ from a free graded presentation of M :

$$\prod_{i \in I} \mathcal{S}(\alpha_i) \xrightarrow{\varphi} \prod_{j \in J} \mathcal{S}(\beta_j) \longrightarrow M \longrightarrow 0.$$

Algorithm

- 1 Compute algebra generators for $(\mathcal{S}_\tau)_0$
- 2 Compute $(\mathcal{S}_\tau)_0$ -generators of $(\mathcal{S}(\gamma)_\tau)_0$, $\gamma \in \{\alpha_i, \beta_j\}$
- 3 Compute $(\varphi_\tau)_0$ using the embedding $P' \subseteq \text{Mon}(\mathcal{S}_\tau)$ by computing representations for $\varphi_\tau(p')$ for all $p' \in P'$.

Presentation of $(M_\tau)_0$

Strategy for computing a presentation of $(M_\tau)_0$ from a free graded presentation of M :

$$\prod_{i \in I} (\mathcal{S}(\alpha_i)_\tau) \xrightarrow{(\varphi_\tau)_0} \prod_{j \in J} (\mathcal{S}(\beta_j)_\tau)_0 \longrightarrow (M_\tau)_0 \longrightarrow 0.$$

Algorithm

- 1 Compute algebra generators for $(\mathcal{S}_\tau)_0$
- 2 Compute $(\mathcal{S}_\tau)_0$ -generators of $(\mathcal{S}(\gamma)_\tau)_0$, $\gamma \in \{\alpha_i, \beta_j\}$
- 3 Compute $(\varphi_\tau)_0$ using the embedding $P' \subseteq \text{Mon}(\mathcal{S}_\tau)$

Coherent sheaves

So $S\text{-grmod}_G^0$ is decidable

Coherent sheaves

So $S\text{-grmod}_G^0$ is decidable and therefore

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

computable abelian!

Coherent sheaves

So $S\text{-grmod}_G^0$ is decidable and therefore

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

computable abelian!

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

Coherent sheaves

So $S\text{-grmod}_G^0$ is decidable and therefore

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

computable abelian!

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

- Homology

Coherent sheaves

So $S\text{-grmod}_G^0$ is decidable and therefore

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

computable abelian!

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

- Homology
- Diagram chases

Coherent sheaves

So $S\text{-grmod}_G^0$ is decidable and therefore

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

computable abelian!

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

- Homology
- Diagram chases
- Spectral sequences

Coherent sheaves

So $S\text{-grmod}_G^0$ is decidable and therefore

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

computable abelian!

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

- Homology
- Diagram chases
- Spectral sequences
- Purity filtration

Coherent sheaves

So $S\text{-grmod}_G^0$ is decidable and therefore

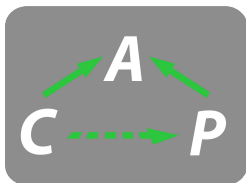
$$S\text{-grmod}_G / S\text{-grmod}_G^0 \cong \mathcal{Coh}(X)$$

computable abelian!

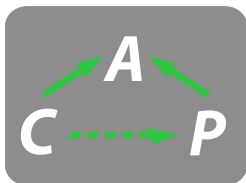
We can apply algorithms for abelian categories to coherent sheaves over normal toric varieties:

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

Workshop: CAP Days 2018

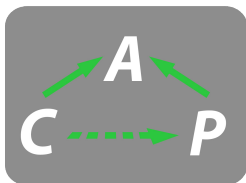


Workshop: CAP Days 2018



August 28 - 31, 2018 at University of Siegen, featuring

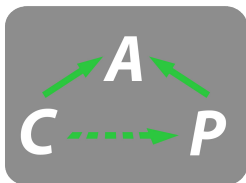
Workshop: CAP Days 2018



August 28 - 31, 2018 at University of Siegen, featuring

- Talks on ongoing research

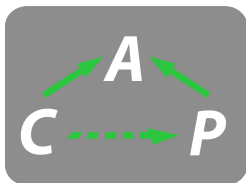
Workshop: CAP Days 2018



August 28 - 31, 2018 at University of Siegen, featuring

- Talks on ongoing research
- Tutorials

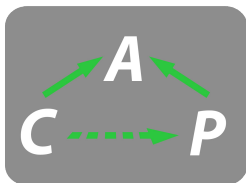
Workshop: CAP Days 2018



August 28 - 31, 2018 at University of Siegen, featuring

- Talks on ongoing research
- Tutorials
- Coding Sprint

Workshop: CAP Days 2018



August 28 - 31, 2018 at University of Siegen, featuring

- Talks on ongoing research
- Tutorials
- Coding Sprint

More information and registration:

<https://homalg-project.github.io/capdays-2018/>