homalg – Constructive Homological Algebra

Mohamed Barakat

RWTH Aachen University

3rd GAP Days
Trondheim, September 14, 2015.

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

The category R-fpmod

- For developing homological algebra constructively we first need to deal with finitely presented modules.

The category R-$\mathbf{fpmod}$

- For developing homological algebra constructively we first need to deal with finitely presented modules.
- A lot of what we want to compute about such modules only depends on their category

$$R\text{-}\mathbf{fpmod} := \begin{cases} \text{Obj:} & \text{finitely presented } R\text{-modules,} \\ \text{Mor:} & \text{their } R\text{-module maps} \end{cases}$$

The category R-$\mathbf{fpmod}$

- For developing homological algebra constructively we first need to deal with finitely presented modules.
- A lot of what we want to compute about such modules only depends on their category

$$R\text{-}\mathbf{fpmod} := \begin{cases} \text{Obj:} & \text{finitely presented } R\text{-modules,} \\ \text{Mor:} & \text{their } R\text{-module maps} \end{cases}$$

up to equivalence.

The algebra of morphisms

How can category theory be helpful in the development of a constructive approach?

How can category theory be helpful in the development of a constructive approach?

Answer:

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

The algebra of morphisms

How can category theory be helpful in the development of a constructive approach?

Answer:

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

The algebra of morphisms

How can category theory be helpful in the development of a constructive approach?

Answer:

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

How can category theory be helpful in the development of a constructive approach?

Answer:

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

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

Equivalence of categories

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

Equivalence of categories

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

Equivalence of categories

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

Linear algebra and matrix theory

Here is a prominent example of this approach.

Here is a prominent example of this approach.

Example

Let k be a field. Then

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

Here is a prominent example of this approach.

Example

Let k be a field. Then

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

$$\simeq$$

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

Here is a prominent example of this approach.

Example

Let k be a field. Then

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

$$\simeq$$

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

Here is a prominent example of this approach.

Example

Let k be a field. Then

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

$$\simeq$$

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

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

A constructive model for R-fpmod

From now on let R be a ring with 1.

From now on let R be a ring with 1.

Definition

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

A constructive model for R-fpmod

From now on let R be a ring with 1.

Definition

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

A constructive model for R-fpmod

From now on let R be a ring with 1.

Definition

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

A constructive model for R-fpmod

From now on let R be a ring with 1.

Definition

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

Example

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

A constructive model for R-fpmod

From now on let R be a ring with 1.

Definition

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

Example

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

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

A constructive model for R-fpmod

From now on let R be a ring with 1.

Definition

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

Example

$$R\textbf{-fpmod} \simeq$$

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

A constructive model for R-fpmod

From now on let R be a ring with 1.

Definition

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

Example

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

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

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

Recall:

Definition

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

Recall:

Definition

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

- finite biproducts exist,

Recall:

Definition

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

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

Recall:

Definition

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

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

Recall:

Definition

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

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

Recall:

Definition

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

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

Definition

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

Example

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

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

The “hidden” existential quantifiers of “kernels”

Example

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

$$\begin{tikzcd} \ker\varphi & & \\ & M \arrow[r, "\varphi"] & N \end{tikzcd}$$

The “hidden” existential quantifiers of “kernels”

Example

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

$$\begin{tikzcd} \ker\varphi \arrow[dr, "\kappa"] & & \\ & M \arrow[r, "\varphi"] & N \end{tikzcd}$$

The "hidden" existential quantifiers of "kernels"

Example

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

$$\begin{tikzcd} \color{blue}{\ker\varphi} \arrow[dr, blue, "\kappa"] \arrow[drr, bend left, "0"] & & \\ & M \arrow[r, "\varphi"] & N \end{tikzcd}$$

Example

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

$$\begin{tikzcd}
\ker\varphi \arrow[dr, "\kappa"] \arrow[drr, bend left, "0"] & & \\
& M \arrow[r, "\varphi"] & N \\
L \arrow[ur, "\tau"] \arrow[urr, bend right, "0"'] & &
\end{tikzcd}$$

The “hidden” existential quantifiers of “kernels”

Example

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

$$
\begin{tikzcd}
\ker\varphi \arrow[dr, "\kappa"] \arrow[drr, bend left, "0"] & & \\
& M \arrow[r, "\varphi"] & N \\
L \arrow[uu, dotted, "\tau/\kappa"] \arrow[ur, "\tau"] \arrow[urr, bend right, "0"'] & &
\end{tikzcd}
$$

$\mathcal{A}$ is a **category**:

1. For any object M there exists an **identity morphism** 1_M.
2. For any two composable morphisms φ, ψ there exists a **composition** $\varphi\psi$.

$\mathcal{A}$ is a category **with zero**

$\mathcal{A}$ is a category **with zero**:

3. There exists a **zero object** 0.
4. For all objects M, N there exists a **zero morphism** 0_{MN}.

$\mathcal{A}$ is an **additive** category

$\mathcal{A}$ is an **additive** category:

5. For all objects M, N there exists an **addition** $(\varphi, \psi) \mapsto \varphi + \psi$ in the ABELian group $\mathrm{Hom}_{\mathcal{A}}(M, N)$.
6. For all objects M, N there exists a **subtraction** $(\varphi, \psi) \mapsto \varphi - \psi$ in the ABELian group $\mathrm{Hom}_{\mathcal{A}}(M, N)$.
7. For all objects A_1, A_2 there exists a **direct sum** $A_1 \oplus A_2$ and projections $\pi_i : A_1 \oplus A_2 \to A_i$ such that
8. for all pairs of morphisms $\varphi_i : M \to A_i$, $i = 1, 2$ there exists a *unique* **product morphism** $\{\varphi_1, \varphi_2\} : M \to A_1 \oplus A_2$ satisfying $\{\varphi_1, \varphi_2\}\pi_i = \varphi_i$.
9. for all pairs of morphisms $\varphi_i : A_i \to M$, $i = 1, 2$ there exists a *unique* **coproduct morphism**[a] $\langle\varphi_1, \varphi_2\rangle : A_1 \oplus A_2 \to M$.

[a] follows from the above axioms [HS97, Prop. II.9.1].

$\mathcal{A}$ is a **pre-Abelian** category:

10. For any morphism $\varphi : M \to N$ there exists a **kernel** $\ker \varphi \xhookrightarrow{\kappa} M$, such that
11. for any morphism $\tau : L \to M$ with $\tau\varphi = 0$ there exists a *unique* **lift** $\tau_0 : L \to \ker \varphi$ of τ along κ, i.e., $\tau_0\kappa = \tau$.
12. For any morphism $\varphi : M \to N$ there exists a **cokernel** $N \overset{\varepsilon}{\twoheadrightarrow} \operatorname{coker} \varphi$, such that
13. for any morphism $\eta : N \to L$ with $\varphi\eta = 0$ there exists a *unique* **colift** $\eta_0 : \operatorname{coker} \varphi \to L$ of η along ε, i.e., $\varepsilon\eta_0 = \eta$.

$\mathcal{A}$ is an ABELian category

$\mathcal{A}$ is an ABELian category:

14. Each mono is a kernel mono.
15. Each epi is a cokernel epi.

Computable rings

Definition

We call a constructive ring **left computable**

Definition

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

Computable rings

Definition

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

- Determining a **syzygy matrix** $\mathtt{S}$ of $\mathtt{A}$:

$$\mathtt{SA} = 0, \forall \mathtt{S}' : \mathtt{S}'\mathtt{A} = 0 \implies \mathtt{S} \geq \mathtt{S}'.$$

Computable rings

Definition

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

- Determining a **syzygy matrix** $\mathtt{S}$ of $\mathtt{A}$:

$$\mathtt{SA} = 0, \forall \mathtt{S}' : \mathtt{S}'\mathtt{A} = 0 \implies \mathtt{S} \geq \mathtt{S}'.$$

- Deciding if $\mathtt{A} \geq \mathtt{B}$, i.e., the solvability of $\mathtt{XA} = \mathtt{B}$

Computable rings

Definition

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

- Determining a **syzygy matrix** $\mathtt{S}$ of $\mathtt{A}$:

$$\mathtt{SA} = 0, \forall \mathtt{S}' : \mathtt{S}'\mathtt{A} = 0 \implies \mathtt{S} \geq \mathtt{S}'.$$

- Deciding if $\mathtt{A} \geq \mathtt{B}$, i.e., the solvability of $\mathtt{XA} = \mathtt{B}$ and in the affirmative case determining a **particular solution** $\mathtt{X}$.

Computable rings

Definition

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

- Determining a **syzygy matrix** $\mathtt{S}$ of $\mathtt{A}$:

$$\mathtt{S}\mathtt{A} = 0, \forall \mathtt{S}' : \mathtt{S}'\mathtt{A} = 0 \implies \mathtt{S} \geq \mathtt{S}'.$$

- Deciding if $\mathtt{A} \geq \mathtt{B}$, i.e., the solvability of $\mathtt{X}\mathtt{A} = \mathtt{B}$ and in the affirmative case determining a **particular solution** $\mathtt{X}$.

Theorem ([BLH11])

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

Submodule membership problem

Rows of the matrices `A` and `B` can be considered as elements of the free module $R^{1\times g}$.

Submodule membership problem

Rows of the matrices `A` and `B` can be considered as elements of the free module $R^{1\times g}$.

- Deciding the solvability of the inhomogeneous linear system $\mathtt{XA} = \mathtt{B}$ for a single row matrix `B`

Submodule membership problem

Rows of the matrices `A` and `B` can be considered as elements of the free module $R^{1\times g}$.

- Deciding the solvability of the inhomogeneous linear system $\mathtt{XA} = \mathtt{B}$ for a single row matrix `B` is thus nothing but the **submodule membership problem** for the submodule generated by the rows of the matrix `A`.

Submodule membership problem

Rows of the matrices A and B can be considered as elements of the free module $R^{1\times g}$.

- Deciding the solvability of the inhomogeneous linear system $\mathtt{XA} = \mathtt{B}$ for a single row matrix B is thus nothing but the **submodule membership problem** for the submodule generated by the rows of the matrix A.
- Finding a particular solution X (in case one exists) solves the submodule membership problem **effectively**.

DecideZeroRows

- $\mathtt{DecideZeroRows}(\mathtt{B}, \mathtt{A})$ returns a matrix $\mathtt{B}'$ (having the same shape as B)

DecideZeroRows

- $\mathtt{DecideZeroRows}(\mathtt{B}, \mathtt{A})$ returns a matrix $\mathtt{B}'$ (having the same shape as $\mathtt{B}$)
  - for which the equation $\mathtt{XA} = \mathtt{B} - \mathtt{B}'$ is solvable

$\mathtt{X} = \mathtt{RightDivide}(\mathtt{B}, \mathtt{A})$

DecideZeroRows

- $\mathtt{DecideZeroRows}(\mathtt{B}, \mathtt{A})$ returns a matrix $\mathtt{B}'$ (having the same shape as $\mathtt{B}$)
  - for which the equation $\mathtt{XA} = \mathtt{B} - \mathtt{B}'$ is solvable
  - and where the i-th row $\mathtt{B}'_i$ is zero iff the equation $\mathtt{xA} = \mathtt{B}_i$ is solvable.

DecideZeroRows

- $\mathtt{DecideZeroRows}(\mathtt{B}, \mathtt{A})$ returns a matrix $\mathtt{B}'$ (having the same shape as $\mathtt{B}$)
  - for which the equation $\mathtt{XA} = \mathtt{B} - \mathtt{B}'$ is solvable
  - and where the i-th row $\mathtt{B}'_i$ is zero iff the equation $\mathtt{xA} = \mathtt{B}_i$ is solvable.

In particular, the equation $\mathtt{XA} = \mathtt{B}$ is solvable iff

$$\mathtt{DecideZeroRows}(\mathtt{B}, \mathtt{A}) = 0.$$

DecideZeroRows

- $\mathtt{DecideZeroRows}(\mathtt{B}, \mathtt{A})$ returns a matrix $\mathtt{B}'$ (having the same shape as $\mathtt{B}$)
  - for which the equation $\mathtt{XA} = \mathtt{B} - \mathtt{B}'$ is solvable
  - and where the i-th row $\mathtt{B}'_i$ is zero iff the equation $\mathtt{xA} = \mathtt{B}_i$ is solvable.

  In particular, the equation $\mathtt{XA} = \mathtt{B}$ is solvable iff

  $$\mathtt{DecideZeroRows}(\mathtt{B}, \mathtt{A}) = \mathtt{0}.$$

- $\mathtt{DecideZeroRowsEffectively}(\mathtt{B}, \mathtt{A})$ computes a matrix $\mathtt{T}$ satisfying $\mathtt{B} + \mathtt{TA} = \mathtt{B}'$, where $\mathtt{B}' = \mathtt{DecideZeroRows}(\mathtt{B}, \mathtt{A})$.

$\mathtt{X} = \mathtt{RightDivide}(\mathtt{B}, \mathtt{A})$

DecideZeroRows

- $\mathtt{DecideZeroRows}(\mathtt{B}, \mathtt{A})$ returns a matrix $\mathtt{B}'$ (having the same shape as $\mathtt{B}$)
  - for which the equation $\mathtt{XA} = \mathtt{B} - \mathtt{B}'$ is solvable
  - and where the i-th row $\mathtt{B}'_i$ is zero iff the equation $\mathtt{xA} = \mathtt{B}_i$ is solvable.

  In particular, the equation $\mathtt{XA} = \mathtt{B}$ is solvable iff

  $$\mathtt{DecideZeroRows}(\mathtt{B}, \mathtt{A}) = 0.$$

- $\mathtt{DecideZeroRowsEffectively}(\mathtt{B}, \mathtt{A})$ computes a matrix $\mathtt{T}$ satisfying $\mathtt{B} + \mathtt{TA} = \mathtt{B}'$, where $\mathtt{B}' = \mathtt{DecideZeroRows}(\mathtt{B}, \mathtt{A})$. In particular, if the equation $\mathtt{XA} = \mathtt{B}$ is solvable then we recover

  $$\mathtt{X} := -\mathtt{T} =: \mathtt{RightDivide}(\mathtt{B}, \mathtt{A}).$$

SyzygiesOfRows and DecideZeroRowsEffectively

Example

```
gap> ?SyzygiesOfRows
```

Example

```
gap> ?SyzygiesOfRows
gap> ?SyzygiesGeneratorsOfRows
```

`SyzygiesOfRows` and `DecideZeroRowsEffectively`

Example

```
gap> ?SyzygiesOfRows
gap> ?SyzygiesGeneratorsOfRows
gap> ?DecideZeroRows
```

SyzygiesOfRows and DecideZeroRowsEffectively

Example

```
gap> ?SyzygiesOfRows
gap> ?SyzygiesGeneratorsOfRows
gap> ?DecideZeroRows
gap> ?DecideZeroRowsEffectively
```

SyzygiesOfRows and DecideZeroRowsEffectively

Example

```
gap> ?SyzygiesOfRows
gap> ?SyzygiesGeneratorsOfRows
gap> ?DecideZeroRows
gap> ?DecideZeroRowsEffectively
gap> ?RightDivide
```

Examples of computable rings

Example (computable rings)

ring	algorithm

a

b

Examples of computable rings

Example (computable rings)

ring	algorithm
a constructive field k	

a

b

Examples of computable rings

Example (computable rings)

ring	algorithm
a constructive field k	
ring of rational integers $\mathbb{Z}$	

a

b

Examples of computable rings

Example (computable rings)

ring	algorithm
a constructive field k	
ring of rational integers $\mathbb{Z}$	
a univariate polynomial ring $k[x]$	

a

b

Examples of computable rings

Example (computable rings)

ring	algorithm
a constructive field k	
ring of rational integers $\mathbb{Z}$	
a univariate polynomial ring $k[x]$	
a polynomial ring[a] $R[x_1, \ldots, x_n]$	

[a] R any of the above rings
[b]

Examples of computable rings

Example (computable rings)

ring	algorithm
a constructive field k	
ring of rational integers $\mathbb{Z}$	
a univariate polynomial ring $k[x]$	
a polynomial ring[a] $R[x_1, \ldots, x_n]$	
many noncommutative rings	

[a] R any of the above rings
[b]

Examples of computable rings

Example (computable rings)

ring	algorithm
a constructive field k	
ring of rational integers $\mathbb{Z}$	
a univariate polynomial ring $k[x]$	
a polynomial ring[a] $R[x_1,\dots,x_n]$	
many noncommutative rings	
$k[x_1,\dots,x_n]_{\langle x_1,\dots,x_n\rangle}$	

[a] R any of the above rings

[b]

Examples of computable rings

Example (computable rings)

ring	algorithm
a constructive field k	
ring of rational integers $\mathbb{Z}$	
a univariate polynomial ring $k[x]$	
a polynomial ring[a] $R[x_1, \ldots, x_n]$	
many noncommutative rings	
$k[x_1, \ldots, x_n]_{\langle x_1, \ldots, x_n \rangle}$	
residue class rings[b]	

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

Examples of computable rings

Example (computable rings)

ring	algorithm
a constructive field k	
ring of rational integers $\mathbb{Z}$	
a univariate polynomial ring $k[x]$	
a polynomial ring[a] $R[x_1, \ldots, x_n]$	
many noncommutative rings	
$k[x_1, \ldots, x_n]_{\langle x_1, \ldots, x_n \rangle}$	
residue class rings[b]	
...	

[a] R any of the above rings

[b] modulo ideals which are f.g. as left resp. right ideals.

Examples of computable rings

Example (computable rings)

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

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

Examples of computable rings

Example (computable rings)

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

[a] R any of the above rings

[b] modulo ideals which are f.g. as left resp. right ideals.

Examples of computable rings

Example (computable rings)

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

[a] R any of the above rings

[b] modulo ideals which are f.g. as left resp. right ideals.

Examples of computable rings

Example (computable rings)

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

[a] R any of the above rings

[b] modulo ideals which are f.g. as left resp. right ideals.

Examples of computable rings

Example (computable rings)

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

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

Examples of computable rings

Example (computable rings)

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

[a] R any of the above rings

[b] modulo ideals which are f.g. as left resp. right ideals.

Examples of computable rings

Example (computable rings)

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

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

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

Exercise

Use `BasisOfRows` to program

- `DecideZeroRows`,
- `DecideZeroRowsEffectively`,
- and `SyzygiesOfRows`.

Exercise

Use `BasisOfRows` to program

- `DecideZeroRows`,
- `DecideZeroRowsEffectively`,
- and `SyzygiesOfRows`.

Hint:

$$\begin{pmatrix} 1 & -X \\ 0 & Y \\ 0 & S \end{pmatrix} \begin{pmatrix} 1 & B & 0 \\ 0 & A & 1 \end{pmatrix} \xrightarrow{\texttt{BasisOfRows}} \begin{pmatrix} 1 & B' & -X \\ 0 & A' & Y \\ 0 & 0 & S \end{pmatrix} = \begin{pmatrix} 1 & -X \\ 0 & Y \\ 0 & S \end{pmatrix} \begin{pmatrix} 1 & B & 0 \\ 0 & A & 1 \end{pmatrix}$$

Some ring constructions in the homalg project

Example (homalg rings)

```
gap> LoadPackage( "RingsForHomalg" );;
```

Some ring constructions in the homalg project

Example (homalg rings)

```
gap> LoadPackage( "RingsForHomalg" );;
gap> ?Ring Constructors
```

Example (homalg rings)

```
gap> LoadPackage( "RingsForHomalg" );;
gap> ?Ring Constructors
gap> Q := HomalgFieldOfRationals( );
Q
```

Example (homalg rings)

```
gap> LoadPackage( "RingsForHomalg" );;
gap> ?Ring Constructors
gap> Q := HomalgFieldOfRationals( );
Q
gap> F2 := HomalgRingOfIntegers( 2 );
GF(2)
```

Example (homalg rings)

```
gap> LoadPackage( "RingsForHomalg" );;
gap> ?Ring Constructors
gap> Q := HomalgFieldOfRationals( );
Q
gap> F2 := HomalgRingOfIntegers( 2 );
GF(2)
gap> F4 := HomalgRingOfIntegers( 2, 2 );
GF(2^2)
```

Example (homalg rings)

```
gap> LoadPackage( "RingsForHomalg" );;
gap> ?Ring Constructors
gap> Q := HomalgFieldOfRationals( );
Q
gap> F2 := HomalgRingOfIntegers( 2 );
GF(2)
gap> F4 := HomalgRingOfIntegers( 2, 2 );
GF(2^2)
gap> ZZ := HomalgRingOfIntegers( );
Z
```

Some ring constructions in the homalg project

Example (homalg rings; SINGULAR needs to be installed)

```
gap> LoadPackage( "RingsForHomalg" );;
gap> ?Ring Constructors
gap> Q := HomalgFieldOfRationals( );
Q
gap> F2 := HomalgRingOfIntegers( 2 );
GF(2)
gap> F4 := HomalgRingOfIntegers( 2, 2 );
GF(2^2)
gap> ZZ := HomalgRingOfIntegers( );
Z
gap> ?Ring Constructions
```

Some ring constructions in the homalg project

Example (homalg rings; SINGULAR needs to be installed)

```
gap> LoadPackage( "RingsForHomalg" );;
gap> ?Ring Constructors
gap> Q := HomalgFieldOfRationals( );
Q
gap> F2 := HomalgRingOfIntegers( 2 );
GF(2)
gap> F4 := HomalgRingOfIntegers( 2, 2 );
GF(2^2)
gap> ZZ := HomalgRingOfIntegers( );
Z
gap> ?Ring Constructions
gap> Q := HomalgFieldOfRationalsInSingular( );
Q
```

Some ring constructions in the homalg project

Example (homalg rings; SINGULAR needs to be installed)

```
gap> LoadPackage( "RingsForHomalg" );;
gap> ?Ring Constructors
gap> Q := HomalgFieldOfRationals( );
Q
gap> F2 := HomalgRingOfIntegers( 2 );
GF(2)
gap> F4 := HomalgRingOfIntegers( 2, 2 );
GF(2^2)
gap> ZZ := HomalgRingOfIntegers( );
Z
gap> ?Ring Constructions
gap> Q := HomalgFieldOfRationalsInSingular( );
Q
gap> F2 := HomalgRingOfIntegersInSingular( 2 );
GF(2)
```

Some ring constructions in the homalg project

Example (homalg rings; SINGULAR needs to be installed)

```
gap> LoadPackage( "RingsForHomalg" );;
gap> ?Ring Constructors
gap> Q := HomalgFieldOfRationals( );
Q
gap> F2 := HomalgRingOfIntegers( 2 );
GF(2)
gap> F4 := HomalgRingOfIntegers( 2, 2 );
GF(2^2)
gap> ZZ := HomalgRingOfIntegers( );
Z
gap> ?Ring Constructions
gap> Q := HomalgFieldOfRationalsInSingular( );
Q
gap> F2 := HomalgRingOfIntegersInSingular( 2 );
GF(2)
gap> F4 := HomalgRingOfIntegersInSingular( 2, 2 );
GF(2^2)
```

Some ring constructions in the homalg project

Example (homalg rings; SINGULAR needs to be installed)

```
gap> LoadPackage( "RingsForHomalg" );;
gap> ?Ring Constructors
gap> Q := HomalgFieldOfRationals( );
Q
gap> F2 := HomalgRingOfIntegers( 2 );
GF(2)
gap> F4 := HomalgRingOfIntegers( 2, 2 );
GF(2^2)
gap> ZZ := HomalgRingOfIntegers( );
Z
gap> ?Ring Constructions
gap> Q := HomalgFieldOfRationalsInSingular( );
Q
gap> F2 := HomalgRingOfIntegersInSingular( 2 );
GF(2)
gap> F4 := HomalgRingOfIntegersInSingular( 2, 2 );
GF(2^2)
gap> ZZ := HomalgRingOfIntegersInSingular( );
Z
```

Some ring constructions in the homalg project

Example (homalg rings; SINGULAR needs to be installed)

```
gap> LoadPackage( "RingsForHomalg" );;
gap> ?Ring Constructors
gap> Q := HomalgFieldOfRationals( );
Q
gap> F2 := HomalgRingOfIntegers( 2 );
GF(2)
gap> F4 := HomalgRingOfIntegers( 2, 2 );
GF(2^2)
gap> ZZ := HomalgRingOfIntegers( );
Z
gap> ?Ring Constructions
gap> Q := HomalgFieldOfRationalsInSingular( );
Q
gap> F2 := HomalgRingOfIntegersInSingular( 2 );
GF(2)
gap> F4 := HomalgRingOfIntegersInSingular( 2, 2 );
GF(2^2)
gap> ZZ := HomalgRingOfIntegersInSingular( );
Z
gap> R := F4 * "x,y,z";
GF(2^2)[x,y,z]
```

Matrix constructions in the homalg project

Example

```
gap> ?HomalgMatrix
```

Example

```
gap> ?HomalgMatrix
gap> ZZ := HomalgRingOfIntegers( );
Z
```

Matrix constructions in the homalg project

Example

```
gap> ?HomalgMatrix
gap> ZZ := HomalgRingOfIntegers( );
Z
gap> m := HomalgMatrix( "[ 1, 2, 3,   4, 5, 6 ]", 2, 3, ZZ );
<A 2 x 3 matrix over an internal ring>
```

Matrix constructions in the homalg project

Example

```
gap> ?HomalgMatrix
gap> ZZ := HomalgRingOfIntegers( );
Z
gap> m := HomalgMatrix( "[ 1, 2, 3,   4, 5, 6 ]", 2, 3, ZZ );
<A 2 x 3 matrix over an internal ring>
gap> m := HomalgMatrix( "[ \
> 1, 2, 3, \
> 4, 5, 6 \
> ]", 2, 3, ZZ );
<A 2 x 3 matrix over an internal ring>
```

Matrix constructions in the homalg project

Example

```
gap> ?HomalgMatrix
gap> ZZ := HomalgRingOfIntegers( );
Z
gap> m := HomalgMatrix( "[ 1, 2, 3,   4, 5, 6 ]", 2, 3, ZZ );
<A 2 x 3 matrix over an internal ring>
gap> m := HomalgMatrix( "[ \
> 1, 2, 3, \
> 4, 5, 6 \
> ]", 2, 3, ZZ );
<A 2 x 3 matrix over an internal ring>
gap> Display( m );
[ [  1,  2,  3 ],
  [  4,  5,  6 ] ]
```

$\mathcal{A}$ is a **pre-Abelian** category:

10. $\boxed{\text{For any morphism } \varphi : M \to N \text{ there exists a **kernel** } \ker\varphi \stackrel{\kappa}{\hookrightarrow} M \text{, such that}}$
11. for any morphism $\tau : L \to M$ with $\tau\varphi = 0$ there exists a *unique* **lift** $\tau_0 : L \to \ker\varphi$ of τ along κ, i.e., $\tau_0\kappa = \tau$.
12. For any morphism $\varphi : M \to N$ there exists a **cokernel** $N \stackrel{\varepsilon}{\twoheadrightarrow} \operatorname{coker}\varphi$, such that
13. for any morphism $\eta : N \to L$ with $\varphi\eta = 0$ there exists a *unique* **colift** $\eta_0 : \operatorname{coker}\varphi \to L$ of η along ε, i.e., $\varepsilon\eta_0 = \eta$.

$\mathtt{S} = \mathtt{SyzygiesOfRows}(\mathtt{A}, \mathtt{N})$

For the stacked matrix $\left(\begin{smallmatrix}\mathtt{A}\\ \mathtt{N}\end{smallmatrix}\right)$ we write

$$\mathtt{SyzygiesOfRows}(\left(\begin{smallmatrix}\mathtt{A}\\ \mathtt{N}\end{smallmatrix}\right)) = \left(\begin{smallmatrix}\mathtt{K} & \mathtt{L}\end{smallmatrix}\right)$$

with $\mathtt{KA} + \mathtt{LN} = \mathtt{0}$ and define[a]

$$\mathtt{SyzygiesOfRows}(\mathtt{A}, \mathtt{N}) := \mathtt{K},$$

for which we need a matrix algorithm `CertainColumns` to extract $\mathtt{K}$.

[a] Again, one can derive more efficient algorithms to compute the relative version of `SyzygiesOfRows`.

How to compute $\ker\varphi \overset{\kappa}{\hookrightarrow} M$ of $\varphi : M \to N$?

To compute the kernel $\ker\varphi \overset{\kappa}{\hookrightarrow} \mathtt{M}$ of a morphism $\varphi : \mathtt{M} \xrightarrow{\mathtt{A}} \mathtt{N}$ we do the following:

How to compute $\ker\varphi \xhookrightarrow{\kappa} M$ of $\varphi : M \to N$?

To compute the kernel $\ker\varphi \xhookrightarrow{\kappa} \mathtt{M}$ of a morphism $\varphi : \mathtt{M} \xrightarrow{\mathtt{A}} \mathtt{N}$ we do the following:

1. First compute

$$\mathtt{K} = \mathtt{SyzygiesOfRows}(\mathtt{A}, \mathtt{N}),$$

   the matrix representing κ.

How to compute $\ker\varphi \overset{\kappa}{\hookrightarrow} M$ of $\varphi : M \to N$?

To compute the kernel $\ker\varphi \overset{\kappa}{\hookrightarrow} \mathtt{M}$ of a morphism $\varphi : \mathtt{M} \xrightarrow{\mathtt{A}} \mathtt{N}$ we do the following:

1. First compute

$$\mathtt{K} = \mathtt{SyzygiesOfRows}(\mathtt{A}, \mathtt{N}),$$

   the matrix representing κ.
2. Then $\ker\varphi$ is presented by the matrix

$$\mathtt{SyzygiesOfRows}(\mathtt{K}, \mathtt{M}).$$

$\mathcal{A}$ is a **pre-Abelian** category:

10. For any morphism $\varphi : M \to N$ there exists a **kernel** $\ker \varphi \stackrel{\kappa}{\hookrightarrow} M$, such that
11. $\boxed{\text{for any morphism } \tau : L \to M \text{ with } \tau\varphi = 0 \text{ there exists a } \textit{unique} \textbf{ lift } \tau_0 : L \to \ker\varphi \text{ of } \tau \text{ along } \kappa, \text{ i.e., } \tau_0\kappa = \tau.}$
12. For any morphism $\varphi : M \to N$ there exists a **cokernel** $N \stackrel{\varepsilon}{\twoheadrightarrow} \operatorname{coker} \varphi$, such that
13. for any morphism $\eta : N \to L$ with $\varphi\eta = 0$ there exists a *unique* **colift** $\eta_0 : \operatorname{coker}\varphi \to L$ of η along ε, i.e., $\varepsilon\eta_0 = \eta$.

How to compute the lift $\tau_0 : L \to K$ of τ along κ?

Let $\kappa : \mathtt{K} \xhookrightarrow{\mathtt{K}} \mathtt{M}$ be the kernel monomorphism and $\tau : \mathtt{L} \xrightarrow{\mathtt{T}} \mathtt{M}$ a morphism with $\tau\varphi = 0$ for $\varphi = \operatorname{coker} \kappa$. Then the matrix

$$\mathtt{X} := \mathtt{RightDivide}(\mathtt{T}, \mathtt{K})$$

represents $\tau_0 : \mathtt{L} \to \mathtt{K}$, the lift of τ along κ.

[1] Cf. [BR08, 3.1.1, case (2)]).

How to compute the lift $\tau_0 : L \to K$ of τ along κ?

Let $\kappa : \mathtt{K} \overset{\mathtt{K}}{\hookrightarrow} \mathtt{M}$ be the kernel monomorphism and $\tau : \mathtt{L} \overset{\mathtt{T}}{\to} \mathtt{M}$ a morphism with $\tau\varphi = 0$ for $\varphi = \operatorname{coker}\kappa$. Then the matrix

$$\mathtt{X} := \mathtt{RightDivide}(\mathtt{T}, \mathtt{K})$$

represents $\tau_0 : \mathtt{L} \to \mathtt{K}$, the lift of τ along κ.

It is an easy exercise[1] to check that X represents a *morphism*.

[1] Cf. [BR08, 3.1.1, case (2)]).

Thank you

- Mohamed Barakat and Markus Lange-Hegermann, *An axiomatic setup for algorithmic homological algebra and an alternative approach to localization*, J. Algebra Appl. **10** (2011), no. 2, 269–293, (`arXiv:1003.1943`). MR 2795737 (2012f:18022)
- Mohamed Barakat and Daniel Robertz, `homalg` – *A meta-package for homological algebra*, J. Algebra Appl. **7** (2008), no. 3, 299–317, (`arXiv:math.AC/0701146`). MR 2431811 (2009f:16010)
- P. J. Hilton and U. Stammbach, *A course in homological algebra*, second ed., Graduate Texts in Mathematics, vol. 4, Springer-Verlag, New York, 1997. MR MR1438546 (97k:18001)