A framework for constructive category theory in computer algebra

Sebastian Posur, Kamal Saleh, Fabian Zickgraf

University of Siegen

July 15, 2019
Part I

Constructive category theory
Abstraction of language
Addition of two numbers:

Data type: int

Data type: float
**Addition of two numbers: Assembly**

<table>
<thead>
<tr>
<th>Data type: int</th>
<th>Data type: float</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
```
**Abstraction of language**

**Addition of two numbers: Assembly**

<table>
<thead>
<tr>
<th>Data type: <strong>int</strong></th>
<th>Data type: <strong>float</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>addi:</strong></td>
<td><strong>addf:</strong></td>
</tr>
<tr>
<td>movl %edi, -4(%rsp)</td>
<td>movss %xmm0, -4(%rsp)</td>
</tr>
<tr>
<td>movl %esi, -8(%rsp)</td>
<td>movss %xmm1, -8(%rsp)</td>
</tr>
<tr>
<td>movl -4(%rsp), %esi</td>
<td>movss -4(%rsp), %xmm0</td>
</tr>
<tr>
<td>addl -8(%rsp), %esi</td>
<td>addss -8(%rsp), %xmm0</td>
</tr>
<tr>
<td>movl %esi, %eax</td>
<td>ret</td>
</tr>
<tr>
<td>ret</td>
<td></td>
</tr>
</tbody>
</table>
Addition of two numbers: C

Data type: int

Data type: float
Addition of two numbers: C

**Data type: int**

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

**Data type: float**
Abstraction of language

Addition of two numbers: C

Data type: \texttt{int}

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

Data type: \texttt{float}

```c
float addf( float a, 
    float b )
{
    return a + b;
}
```
Abstraction of language

Addition of two numbers: GAP or Julia

Data type: \texttt{int}

Data type: \texttt{float}
Addition of two numbers: GAP or Julia

**Data type: int**

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

**Data type: float**

```plaintext
def function(a, b)
    return a + b;
end;
```
Addition of two numbers: GAP or Julia

**Data type: int**

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

**Data type: float**

```plaintext
function( a, b )
  return a + b;
end;
```
Abstraction of language

Addition of two numbers: GAP or Julia

Data type: \texttt{int, float}

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

High language leads to generic code!
Addition of two numbers: GAP or Julia

Data type: `int`, `float`

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

High language leads to generic code!
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

\[
x_1 v_1 + x_2 v_2 = y_1 w_1 + y_2 w_2
\]

Ideals of \( \langle x \rangle, \langle y \rangle \leq Z \):

Euclidean algorithm:

\[
\langle \text{lcm}(x, y) \rangle
\]

Generic algorithm for both cases?

Category theory!
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

\[ x_1 v_1 + x_2 v_2 = y_1 w_1 + y_2 w_2 \]
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
\[
x_1 v_1 + x_2 v_2 = y_1 w_1 + y_2 w_2
\]

Ideals of \( \mathbb{Z} \)
Computing the intersection of two subobjects

Vector spaces

\[ \langle v_1, v_2 \rangle, \langle w_1, w_2 \rangle \leq V: \]

Solution of

\[
\begin{align*}
x_1 v_1 + x_2 v_2 &= y_1 w_1 + y_2 w_2
\end{align*}
\]

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
\[ x_1 v_1 + x_2 v_2 = y_1 w_1 + y_2 w_2 \]

Ideals of \( \mathbb{Z} \)
\[ \langle x \rangle, \langle y \rangle \leq \mathbb{Z} : \]
Euclidean algorithm:
\[ \langle \text{lcm}(x, y) \rangle \]
Computing the intersection of two subobjects

<table>
<thead>
<tr>
<th>Vector spaces</th>
<th>Ideals of ( \mathbb{Z} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \langle v_1, v_2 \rangle, \langle w_1, w_2 \rangle \leq V: )</td>
<td></td>
</tr>
<tr>
<td>Solution of</td>
<td></td>
</tr>
<tr>
<td>[ x_1 v_1 + x_2 v_2 = y_1 w_1 + y_2 w_2 ]</td>
<td></td>
</tr>
<tr>
<td>Euclidean algorithm:</td>
<td></td>
</tr>
<tr>
<td>[ \langle \text{lcm} (x, y) \rangle ]</td>
<td></td>
</tr>
</tbody>
</table>

Generic algorithm for both cases?
Computing the intersection of two subobjects

**Vector spaces**

\[ \langle v_1, v_2 \rangle, \langle w_1, w_2 \rangle \leq V: \]

Solution of

\[
x_1 v_1 + x_2 v_2 = y_1 w_1 + y_2 w_2
\]

**Ideals of \( \mathbb{Z} \)**

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

Euclidean algorithm:

\[ \langle \text{lcm} (x, y) \rangle \]

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

CAP implements a categorical programming language.
Category theory abstracts mathematical structures,
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, Programming**
Category theory abstracts mathematical structures, defines a *language* to formulate theorems and algorithms for different structures *at the same time*.

**CAP - Categories, Algorithms, Programming**

CAP implements a **categorical programming language**.
A category $\mathcal{C}$ contains the following data:
A category $C$ contains the following data:

- $\text{Obj}_C$
A category $C$ contains the following data:

- $\text{Obj}_C$
- $\text{Hom}_C(A, B)$
Definition

A category $C$ contains the following data:

- $\text{Obj}_C$
- $\text{Hom}_C(A, B)$

![Diagram](https://via.placeholder.com/150)
A category $C$ contains the following data:

- $\text{Obj}_C$
- $\text{Hom}_C(A, B)$
A category $\mathcal{C}$ contains the following data:

- $\text{Obj}_\mathcal{C}$
- $\text{Hom}_\mathcal{C}(A, B)$
- $\cdot : \text{Hom}_\mathcal{C}(A, B) \times \text{Hom}_\mathcal{C}(B, C) \to \text{Hom}_\mathcal{C}(A, C)$ (assoc.)

Diagram:

```
A  →  B  →  C
```
A category $\mathcal{C}$ contains the following data:

- $\text{Obj}_\mathcal{C}$
- $\text{Hom}_\mathcal{C}(A, B)$
- $\cdot : \text{Hom}_\mathcal{C}(A, B) \times \text{Hom}_\mathcal{C}(B, C) \to \text{Hom}_\mathcal{C}(A, C)$ (assoc.)
Definition

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

- $\text{Obj}_\mathcal{C}$
- $\text{Hom}_\mathcal{C}(A, B)$
- $\cdot : \text{Hom}_\mathcal{C}(A, B) \times \text{Hom}_\mathcal{C}(B, C) \rightarrow \text{Hom}_\mathcal{C}(A, C)$ (assoc.)
- Neutral elements: $\text{id}_A \in \text{Hom}_\mathcal{C}(A, A)$
A category $\mathcal{C}$ contains the following data:

- $\text{Obj}_\mathcal{C}$
- $\text{Hom}_\mathcal{C}(A, B)$
- $\cdot : \text{Hom}_\mathcal{C}(A, B) \times \text{Hom}_\mathcal{C}(B, C) \rightarrow \text{Hom}_\mathcal{C}(A, C)$ (assoc.)
- Neutral elements: $\text{id}_A \in \text{Hom}_\mathcal{C}(A, A)$
A category $\mathcal{C}$ contains the following data:

- $\text{Obj}_\mathcal{C}$
- $\text{Hom}_\mathcal{C}(A, B)$
- $\cdot : \text{Hom}_\mathcal{C}(A, B) \times \text{Hom}_\mathcal{C}(B, C) \to \text{Hom}_\mathcal{C}(A, C)$ (assoc.)
- Neutral elements: $\text{id}_A \in \text{Hom}_\mathcal{C}(A, A)$
Finite dimensional vector spaces

Let $k$ be a field (e.g., $k = \mathbb{Q}$).
Finite dimensional vector spaces

Let $k$ be a field (e.g., $k = \mathbb{Q}$).

Example: $k$-vec

- Obj := finite dimensional $k$-vector spaces
Finite dimensional vector spaces

Let $k$ be a field (e.g., $k = \mathbb{Q}$).

**Example: $k$-vec**

- $\text{Obj} := \text{finite dimensional } k\text{-vector spaces}$
- $\text{Hom}(V, W) := k\text{-linear maps } V \rightarrow W$
Finite dimensional vector spaces

Let $k$ be a field (e.g., $k = \mathbb{Q}$).

**Example: $k$-vec**

- Obj := finite dimensional $k$-vector spaces
- $\text{Hom}(V, W) := k$-linear maps $V \to W$

**Example: matrices**

- Obj := $\mathbb{N}_0$
Let $k$ be a field (e.g., $k = \mathbb{Q}$).

**Example: $k$-vec**

- Obj := finite dimensional $k$-vector spaces
- Hom($V$, $W$) := $k$-linear maps $V \to W$

**Example: matrices**

- Obj := $\mathbb{N}_0$
- Hom($n$, $m$) := $k^{n \times m}$
Finite dimensional vector spaces

Let $k$ be a field (e.g., $k = \mathbb{Q}$).

**Example: $k$-vec**
- Obj := finite dimensional $k$-vector spaces
- Hom($V, W$) := $k$-linear maps $V \to W$

**Example: matrices**
- Obj := $\mathbb{N}_0$
- Hom($n, m$) := $k^{n \times m}$
Let $k$ be a field (e.g., $k = \mathbb{Q}$).

**Example: $k$-vec**

- Obj $:= \text{finite dimensional } k\text{-vector spaces}$
- Hom($V, W$) $:= k\text{-linear maps } V \to W$

**Example: matrices (computer friendly model)**

- Obj $:= \mathbb{N}_0$
- Hom($n, m$) $:= k^{n \times m}$
Finite dimensional vector spaces

Let \( k \) be a field (e.g., \( k = \mathbb{Q} \)).

**Example: \( k \)-vec**

- Obj := finite dimensional \( k \)-vector spaces
- \( \text{Hom}(V, W) := k \)-linear maps \( V \rightarrow W \)

**Example: matrices (computerfriendly model)**

- Obj := \( \mathbb{N}_0 \)
- \( \text{Hom}(n, m) := k^{n \times m} \)

We denote this category by \( \text{Rows}_k \).
A category becomes computable through data structures for objects and morphisms, algorithms to compute the composition of morphisms and identity morphisms of objects.

Example: matrices

\[
\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 11 \\ 1 \end{pmatrix}
\]
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.

Example: matrices

\[
\begin{pmatrix}
1 & 2 \\
1 & 2
\end{pmatrix}
\begin{pmatrix}
3 & 4 \\
3 & 4
\end{pmatrix}
\begin{pmatrix}
1 & 2 \\
1 & 2
\end{pmatrix}
\cdot
\begin{pmatrix}
3 & 4 \\
3 & 4
\end{pmatrix}
= \begin{pmatrix}11 \\
11\end{pmatrix}
\begin{pmatrix}1 & 0 \\
1 & 0\end{pmatrix}
\begin{pmatrix}1 & 0 \\
1 & 0\end{pmatrix}
\begin{pmatrix}1 & 2 \\
1 & 2\end{pmatrix}\]
A category becomes computable through
- data structures for *objects* and *morphisms*,

Example:

\[
\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 11 & 12 \\ 13 & 14 \end{pmatrix}
\]
Computable categories

A category becomes computable through
- data structures for objects and morphisms,
- algorithms to compute the composition of morphisms
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.

Example: matrices
A category becomes computable through
- data structures for *objects* and *morphisms*,
- algorithms to compute the *composition* of morphisms and *identity morphisms* of objects.

Example: matrices
A category becomes computable through
- data structures for objects and morphisms,
- algorithms to compute the composition of morphisms and identity morphisms of objects.

Example: matrices

\[
\begin{pmatrix}
1 & 2 & 1 \\
\end{pmatrix}
\]
A category becomes computable through
- data structures for *objects* and *morphisms*,
- algorithms to compute the *composition* of morphisms and *identity morphisms* of objects.

Example: matrices

\[
\begin{pmatrix}
1 & 2 \\
1 & 1
\end{pmatrix}
\]
A category becomes computable through
- data structures for *objects* and *morphisms*,
- algorithms to compute the *composition* of morphisms and *identity morphisms* of objects.

**Example: matrices**

\[
\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 11 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 1 & 0 \end{pmatrix}
\]
A category becomes computable through

- data structures for objects and morphisms,
- algorithms to compute the composition of morphisms and identity morphisms of objects.

Example: matrices

\[
\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}
\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} =
\begin{pmatrix} 11 & 23 \\ 13 & 24 \end{pmatrix}
\]
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.

**Example: matrices**

\[
\begin{pmatrix}
1 & 2 \\
\end{pmatrix}
\begin{pmatrix}
1 & 2 \\
\end{pmatrix}
\begin{pmatrix}
3 \\
4 \\
\end{pmatrix}
= 
\begin{pmatrix}
11 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
1 & 2 \\
\end{pmatrix}
\begin{pmatrix}
1 & 2 \\
\end{pmatrix}
\begin{pmatrix}
3 \\
4 \\
\end{pmatrix}
= 
\begin{pmatrix}
11 \\
\end{pmatrix}
\]
A category becomes computable through
- data structures for *objects* and *morphisms*,
- algorithms to compute the *composition* of morphisms and *identity morphisms* of objects.

**Example: matrices**

\[
\begin{pmatrix}
1 & 2 \\
\end{pmatrix} \cdot 
\begin{pmatrix}
3 \\
4 \\
\end{pmatrix} =
\begin{pmatrix}
11 \\
\end{pmatrix}
\]
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.

Example: matrices

\[
\begin{pmatrix}
1 & 2 \\
\end{pmatrix}
\begin{pmatrix}
3 \\
4 \\
\end{pmatrix} = 
\begin{pmatrix}
11 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
1 & 2 \\
\end{pmatrix} \begin{pmatrix}
1 & 0 \\
0 & 1 \\
\end{pmatrix}
\begin{pmatrix}
3 \\
4 \\
\end{pmatrix} = 
\begin{pmatrix}
11 \\
\end{pmatrix}
\]

A category becomes computable through
- data structures for *objects* and *morphisms*,
- algorithms to compute the *composition* of morphisms and *identity morphisms* of objects.

**Example: matrices**

\[
\begin{pmatrix}
1 & 2 \\
3 & 4
\end{pmatrix}
\begin{pmatrix}
1 & 0 \\
0 & 1
\end{pmatrix}
\begin{pmatrix}
1 & 2 \\
3 & 4
\end{pmatrix}
= \begin{pmatrix}
11
\end{pmatrix}
\]
Let $R$ be a ring. Finitely presented $R$-modules form a category.
Let $R$ be a ring. Finitely presented $R$-modules form a category

$\text{mod}_R$
Let $R$ be a ring. Finitely presented $R$-modules form a category

$$\text{mod}_R$$

with $R$-linear maps as morphisms.
Let $R$ be a ring. Finitely presented $R$-modules form a category

$$\text{mod}_R$$

with $R$-linear maps as morphisms.

**Computer-friendly model?**
Data structures: objects

\[ \mathbb{Z}^{1 \times 3} \]

\langle \text{Rows of } \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \rangle
Data structures: objects

\[ \mathbb{Z}^{1 \times 3} \]

\[
\begin{pmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{pmatrix}
\]
Data structures: objects

\[
\mathbb{Z}^{1 \times 3} \\
\begin{pmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{pmatrix}
\]
Idea: a matrix $M \in R^{m \times n}$ can represent the module $\frac{R^{1 \times n}}{\langle M \rangle}$. 
Data structures: objects

$$\mathbb{Z}^{1 \times 3}$$

$$\begin{pmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{pmatrix}$$

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

Objects

$$\text{Obj}_{\text{pres}_R} := \bigcup_{m, n \in \mathbb{N}_0} R^{m \times n}$$
Given: \( M \in \mathbb{R}^{m \times n} \) and \( M' \in \mathbb{R}^{m' \times n'} \).
Given: $M \in R^{m\times n}$ and $M' \in R^{m'\times n'}$. 

\[
\begin{array}{c}
R^{1\times n} \quad \langle M \rangle \\
R^{1\times n'} \quad \langle M' \rangle
\end{array}
\]
Given: $M \in R^{m \times n}$ and $M' \in R^{m' \times n'}$.

$\langle M \rangle \xrightarrow{\text{Hom}} \langle M' \rangle$
Given: \( M \in R^{m \times n} \) and \( M' \in R^{m' \times n'} \).

\[
\begin{array}{c}
\mathbb{R}^{1 \times n} \langle M \rangle \\
\mathbb{R}^{1 \times n'} \langle M' \rangle
\end{array}
\]

\[e_i\]
Given: \( M \in R^{m \times n} \) and \( M' \in R^{m' \times n'} \).

\[
\begin{align*}
R_1^{\times n} & \langle M \rangle \quad \longrightarrow \quad R_1^{\times n'} \langle M' \rangle \\
\bar{e}_i & \quad \longmapsto \quad \bar{r}_i
\end{align*}
\]
Data structures: morphisms

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

$$
\begin{pmatrix}
  r_1 \\
  \vdots \\
  r_n 
\end{pmatrix}
$$

$R_1 \times n \langle M \rangle \rightarrow R_1 \times n' \langle M' \rangle$

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

$$\begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix}$$

$R^{1 \times n} \langle M \rangle \xrightarrow{\overline{e_i}} R^{1 \times n'} \langle M' \rangle$

$\overline{e_i} \leftrightarrow \overline{r_i}$

$\text{Hom}_{\text{pres}_R}(M, M') :=$
Given: $M \in R^{m \times n}$ and $M' \in R^{m' \times n'}$.

$$A := \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix}$$

$$\begin{array}{c}
R^{1 \times n} \\
\langle M \rangle
\end{array} \xrightarrow{A} \begin{array}{c}
R^{1 \times n'} \\
\langle M' \rangle
\end{array}$$

$$e_i \mapsto r_i$$

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

$$A \in R^{n \times n'}$$
Data structures: morphisms

Given: \( M \in \mathbb{R}^{m \times n} \) and \( M' \in \mathbb{R}^{m' \times n'} \).

\[
A := \begin{pmatrix}
    r_1 \\
    \vdots \\
    r_n
\end{pmatrix}
\]

\[
\hom_{\mathbb{R}}(M, M') := A \in \mathbb{R}^{n \times n'} \quad \text{such that}
\]
Data structures: morphisms

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

\[
A := \begin{pmatrix}
    r_1 \\
    \vdots \\
    r_n
\end{pmatrix}
\]

\[
\begin{array}{c}
R^{1 \times n} \\
\langle M \rangle
\end{array} \xrightarrow{e_i} \begin{array}{c}
R^{1 \times n'} \\
\langle M' \rangle
\end{array}
\]

\[
\overline{e_i} \rightarrow \overline{r_i}
\]

\[
\text{Hom}_{\text{fres}_R}(M, M') :=
\]

\[
A \in R^{n \times n'} \quad \text{such that} \quad \{\text{Rows of } M \cdot A\} \subseteq \langle M' \rangle
\]
Given: \( M \in R^{m \times n} \) and \( M' \in R^{m' \times n'} \).

\[
A := \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix}
\]

\[
\begin{array}{c}
R^{1 \times n} / \langle M \rangle \\
\overline{e_i} \\
\end{array} \rightarrow \\
\begin{array}{c}
R^{1 \times n'} / \langle M' \rangle \\
\overline{r_i} \\
\end{array}
\]

\[
\text{Hom}_{\text{fpres}}(M, M') :=
\]

\[
A \in R^{n \times n'} \text{ such that } \exists X \in R^{m \times m'} : M \cdot A = X \cdot M'
\]
Given: \( M \in R^{m \times n} \) and \( M' \in R^{m' \times n'} \).

\[
A := \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix}
\]

\[
\begin{array}{c}
R^{1 \times n} \\
\langle M \rangle
\end{array} \rightarrow
\begin{array}{c}
R^{1 \times n'} \\
\langle M' \rangle
\end{array}
\]

\[ \overline{e_i} \rightarrow \overline{r_i} \]

\[
\text{Hom}_{\text{fpres}}(M, M') := A \in R^{n \times n'} \text{ such that } \exists X \in R^{m \times m'} : M \cdot A = X \cdot M'
\]

A defines the 0 morphism
Given: \( M \in R^{m \times n} \) and \( M' \in R^{m' \times n'} \).

\[
A := \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix}
\]

\[ R^{1 \times n} \langle M \rangle \rightarrow R^{1 \times n'} \langle M' \rangle \]

\[ \overline{e_i} \rightarrow \overline{r_i} \]

\[
\text{Hom}_{f\text{pres}_R}(M, M') :=
\]

\[
A \in R^{n \times n'} \quad \text{such that} \quad \exists X \in R^{m \times m'} : M \cdot A = X \cdot M'
\]

A defines the 0 morphism iff \( \exists X \in R^{n \times m'} : A = X \cdot M' \).
Given: $M \in \mathbb{R}^{m\times n}$ and $M' \in \mathbb{R}^{m'\times n'}$.

$$A := \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix}$$

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

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

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

$$A := \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix}$$

$\langle M \rangle \xrightarrow{\mathbb{R}^{1 \times n}} \mathbb{R}^{1 \times n'}_{\langle M' \rangle}$

$e_i \mapsto r_i$

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

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

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

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

$$A := \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix}$$

$$\xymatrix{ R^{1 \times n} \ar[r]^{\langle M \rangle} & R^{1 \times n'} \ar[l]_{\langle M' \rangle} }$$

$$e_i \mapsto r_i$$

**Hom$_{fpres_R}(M, M') :=$**

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

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

$\leadsto$ algorithmic requirements for $R$ (for $R = \mathbb{Z}$, use Smith normal forms)
The language of category theory

$k$-vec and $\text{mod}_\mathbb{Z}$ are examples of **abelian categories**.
The language of category theory

$k\text{-vec}$ and $\text{mod}_\mathbb{Z}$ are examples of **abelian categories**.

Some categorical operations in abelian categories

- $\oplus: \text{Obj} \times \text{Obj} \to \text{Obj}$
- $\pm: \text{Hom}(A, B) \times \text{Hom}(A, B) \to \text{Hom}(A, B)$
- $\ker: \text{Hom}(A, B) \to \text{Obj}$
The language of category theory

$k$-vec and $\text{mod}_\mathbb{Z}$ are examples of **abelian categories**.

Some categorical operations in abelian categories

- $\oplus : \text{Obj} \times \text{Obj} \rightarrow \text{Obj}$
The language of category theory

$k$-vec and $\text{mod}_\mathbb{Z}$ are examples of abelian categories.

Some categorical operations in abelian categories

- $\oplus : \text{Obj} \times \text{Obj} \rightarrow \text{Obj}$
- $+, - : \text{Hom}(A, B) \times \text{Hom}(A, B) \rightarrow \text{Hom}(A, B)$
$k$-vec and $\text{mod}_\mathbb{Z}$ are examples of **abelian categories**.

Some categorical operations in abelian categories

- $\oplus : \text{Obj} \times \text{Obj} \rightarrow \text{Obj}$
- $+, - : \text{Hom}(A, B) \times \text{Hom}(A, B) \rightarrow \text{Hom}(A, B)$
- $\ker : \text{Hom}(A, B) \rightarrow \text{Obj}$
*k*-*vec* and $\text{mod}_\mathbb{Z}$ are examples of **abelian categories**.

Some categorical operations in abelian categories

- $\oplus : \text{Obj} \times \text{Obj} \rightarrow \text{Obj}$
- $+, - : \text{Hom}(A, B) \times \text{Hom}(A, B) \rightarrow \text{Hom}(A, B)$
- $\ker : \text{Hom}(A, B) \rightarrow \text{Obj}$
- ...
\( k\)-\texttt{vec} and \(\text{mod}_\mathbb{Z} \) are examples of \textbf{abelian categories}.

Some categorical operations in abelian categories:

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

---

To fully describe the kernel of $\varphi$, one needs an object $\text{KernelObject}(\varphi)$, its embedding $\kappa = \text{KernelEmbedding}(\varphi)$, and for every test morphism $\tau$ a unique morphism $\lambda = \text{KernelLift}(\varphi, \tau)$, such that $A \xrightarrow{\kappa} B \xleftarrow{\varphi} \text{KernelObject}(\varphi) \xrightarrow{\lambda} \tau$. 

---

**Implementation of the kernel**

---

Posur, Saleh, Zickgraf (Siegen)  
CAP  
July 15, 2019  
13/65
Let $\varphi \in \text{Hom}(A, B)$. 

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

... one needs an object $\text{KernelObject}(\varphi)$,
Let \( \varphi \in \text{Hom}(A, B) \). To fully describe the kernel of \( \varphi \) ... 

... one needs an object \( \text{KernelObject}(\varphi) \), its embedding \( \kappa = \text{KernelEmbedding}(\varphi) \),
Let $\varphi \in \text{Hom}(A, B)$. To fully describe the kernel of $\varphi$ . . .

. . . one needs an object $\text{KernelObject}(\varphi)$, its embedding $\kappa = \text{KernelEmbedding}(\varphi)$, and for every test morphism $\tau$
Let $\varphi \in \text{Hom}(A, B)$. To fully describe the kernel of $\varphi$ ... 

... one needs an object $\text{KernelObject}(\varphi)$, its embedding $\kappa = \text{KernelEmbedding}(\varphi)$, and for every test morphism $\tau$ a unique morphism $\lambda = \text{KernelLift}(\varphi, \tau)$.
Let \( \varphi \in \text{Hom}(A, B) \). To fully describe the kernel of \( \varphi \) ... one needs an object \( \text{KernelObject}(\varphi) \), its embedding \( \kappa = \text{KernelEmbedding}(\varphi) \), and for every test morphism \( \tau \) a unique morphism \( \lambda = \text{KernelLift}(\varphi, \tau) \), such that
Obj := \mathbb{N}_0, \text{Hom}(m, n) := \mathbb{Q}^{m \times n}
Implementation of the kernel: matrices

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

KernelObject(\varphi)

\begin{align*}
A & \xrightarrow{\varphi} B \\
\end{align*}
Obj := \mathbb{N}_0, \text{Hom} (m, n) := \mathbb{Q}^{m \times n}

\text{Compute}

- \text{KernelObject}(\varphi) \text{ as } A - \text{rank}(\varphi)
Obj := $\mathbb{N}_0$, Hom $(m, n) := \mathbb{Q}^{m \times n}$

Compute

KernelObject($\varphi$) as $A - \text{rank}(\varphi)$
Obj := $\mathbb{N}_0$, Hom $(m, n) := \mathbb{Q}^{m \times n}$

Compute
- KernelObject($\varphi$) as $A - \text{rank}(\varphi)$
- $\kappa$ by solving $X \cdot \varphi = 0$
Implementation of the kernel: matrices

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

kernelObject(\(\phi\))

Compute:
- \textbf{KernelObject(\(\phi\)) as} \(A - \text{rank}(\phi)\)
- \(\kappa\) by solving \(X \cdot \phi = 0\)
Obj := \mathbb{N}_0, \text{Hom}(m, n) := \mathbb{Q}^{m \times n}

Compute

- \text{KernelObject}(\varphi) \text{ as } A - \text{rank}(\varphi)
- \kappa \text{ by solving } X \cdot \varphi = 0
Implementation of the kernel: matrices

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

**KernelObject(\(\varphi\))**

- Compute \(\text{KernelObject}(\varphi)\) as \(A - \text{rank}(\varphi)\)
- \(\kappa\) by solving \(X \cdot \varphi = 0\)
- \(\lambda\) by solving \(X \cdot \kappa = \tau\)
Given a diagram of abelian groups:
Given a diagram of abelian groups:
Given a diagram of abelian groups:
Given a diagram of abelian groups:

\[ x \in \ker A \rightarrow A' \rightarrow B' \]

\[ \ker \rightarrow A \rightarrow B \]

\[ \alpha \]

\[ \text{Posur, Saleh, Zickgraf (Siegen)} \]
Given a diagram of abelian groups:

\[ x \in \ker \quad \longrightarrow \quad x \in A' \quad \longrightarrow \quad B' \]

\[ \ker \quad \longrightarrow \quad A \quad \longrightarrow \quad B \]

\[ \alpha \downarrow \quad \alpha \downarrow \]
Given a diagram of abelian groups:

\[
x \in \ker \quad \xrightarrow{\alpha} \quad x \in A' \quad \xrightarrow{\alpha} \quad B'
\]

\[
\ker \quad \xrightarrow{\alpha(x)} \quad \alpha(x) \in A \quad \xrightarrow{\alpha} \quad B
\]
Given a diagram of abelian groups:

\[ x \in \ker \longrightarrow x \in A' \longrightarrow B' \]

\[ \ker \quad \alpha \quad \alpha(x) \in A \longrightarrow 0 \in B \]
Given a diagram of abelian groups:
Given a diagram of abelian groups:

\[ \begin{align*}
    x & \in \ker \\
    \alpha(x) & \in \ker \\
    x & \in A' \\
    \alpha(x) & \in A \\
    0 & \in B
\end{align*} \]
The same example in the language of category theory:

\[
\text{ker} \xrightarrow{\kappa'} A' \xrightarrow{\varphi} B' \\
\downarrow \alpha \\
\text{ker} \xrightarrow{} A \xrightarrow{} B
\]
The same example in the language of category theory:
The language of category theory

The same example in the language of category theory:

\[ \ker \rightarrow A' \quad \xrightarrow{\kappa'} \quad B' \]
\[ \ker \rightarrow A \quad \xrightarrow{\alpha} \quad B \]
\[ \phi \]

\[ = \]
The language of category theory

The same example in the language of category theory:

\[
\text{ker} \xrightarrow{k'} A' \xrightarrow{\alpha} B' \\
\text{ker} \xrightarrow{\kappa'} A \xrightarrow{\varphi} B
\]

\[\Downarrow = \kappa' \cdot \alpha\]
The language of category theory

The same example in the language of category theory:

\[
\begin{array}{c}
\text{ker} \quad \xrightarrow{\kappa'} \quad A' \quad \xrightarrow{\varphi} \quad B' \\
\text{ker} \quad \xrightarrow{\alpha} \quad A \quad \xrightarrow{\kappa' \cdot \alpha} \quad B
\end{array}
\]

\[
\Downarrow = \text{KernelLift}(\varphi, \kappa' \cdot \alpha)
\]
The language of category theory

The same example in the language of category theory:

\[ \text{ker} \xrightarrow{\kappa'} A' \xrightarrow{\varphi} B' \]

\[ \text{ker} \xrightarrow{A} B \]

\[ \Downarrow = \text{KernelLift}(\varphi, \kappa' \cdot \alpha) \]

KernelObjectFunctorial
CAP - Categories, Algorithms, Programming
CAP is a framework to implement computable categories and provides specifications of lots of basic operations from category theory, a derivation mechanism that automatically installs lots of basic operations for the user, and higher generic algorithms based on basic categorical operations.
CAP is a framework to implement computable categories and provides:
- specifications of lots of basic operations from category theory,
CAP is a framework to implement computable categories and provides

- specifications of lots of basic operations from category theory,
- a derivation mechanism that automatically installs lots of basic operations for the user,
Features of CAP

**CAP - Categories, Algorithms, Programming**

CAP is a framework to implement computable categories and provides:

- specifications of lots of basic operations from category theory,
- a derivation mechanism that automatically installs lots of basic operations for the user,
- higher generic algorithms based on basic categorical operations.
Computing the intersection

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

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

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

Computing the intersection

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

\begin{align*}
M_1 & \xrightarrow{\iota_1} N \\
M_2 & \xrightarrow{\iota_2} N
\end{align*}
Computing the intersection

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

\[ \pi_1 \cdot \iota_1 - \pi_2 \cdot \iota_2 \]

\[ \kappa \]

\[ \gamma \]
Computing the intersection

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

\[
\begin{align*}
\pi_1 & : M_1 \to M_1 \oplus M_2 \\
\iota_1 & : M_1 \oplus M_2 \to M_1 \\
\iota_2 & : M_1 \oplus M_2 \to N \\
\phi & : \pi_1 \cdot \iota_1 - \pi_2 \cdot \iota_2 \\
\kappa & : \text{Kernel Embedding}(\phi) \\
\gamma & : \kappa \cdot \pi_1 \cdot \iota_1
\end{align*}
\]
Let $M_1 \hookrightarrow N$ and $M_2 \hookrightarrow N$ subobjects in an abelian category. Compute their intersection $\gamma : M_1 \cap M_2 \hookrightarrow N$. 

\[\begin{align*} 
\pi_1 : M_1 &\longrightarrow M_1 \oplus M_2 \\
\pi_2 : M_2 &\longrightarrow M_1 \oplus M_2 \\
\iota_1 &\longrightarrow M_1 \oplus M_2 \\
\iota_2 &\longrightarrow M_1 \oplus M_2 \\
\end{align*}\]
Computing the intersection

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

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

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

\[ \pi_i := \text{ProjectionInFactorOfDirectSum} \left( (M_1, M_2), i \right), \ i = 1, 2 \]
Computing the intersection

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

\[ \pi_i := \text{ProjectionInFactorOfDirectSum}((M_1, M_2), i), \ i = 1, 2 \]
\[ \varphi := \pi_1 \cdot \iota_1 - \pi_2 \cdot \iota_2 \]
Let $M_1 \hookrightarrow N$ and $M_2 \hookrightarrow N$ subobjects in an abelian category. Compute their intersection $\gamma : M_1 \cap M_2 \hookrightarrow N$.

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

\[ \varphi := \pi_1 \cdot \iota_1 - \pi_2 \cdot \iota_2 \]
Computing the intersection

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

\[ \kappa : M_1 \cap M_2 \hookrightarrow M_1 \oplus M_2 \]

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

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

\[ \pi_i := \text{ProjectionInFactorOfDirectSum} \left( (M_1, M_2), i \right), \quad i = 1, 2 \]
\[ \varphi := \pi_1 \cdot \iota_1 - \pi_2 \cdot \iota_2 \]
\[ \kappa := \text{KernelEmbedding} \left( \varphi \right) \]
Computing the intersection

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

\[ \pi_i := \text{ProjectionInFactorOfDirectSum} \left( (M_1, M_2), i \right), \quad i = 1, 2 \]
\[ \varphi := \pi_1 \cdot \iota_1 - \pi_2 \cdot \iota_2 \]
\[ \kappa := \text{KernelEmbedding} \left( \varphi \right) \]
\[ \gamma := \kappa \cdot \pi_1 \cdot \iota_1 \]
\[ \pi_i := \text{ProjectionInFactorOfDirectSum}((M_1, M_2), i), \quad i = 1, 2 \]

\[ \varphi := \pi_1 \cdot \iota_1 - \pi_2 \cdot \iota_2 \]

\[ \kappa := \text{KernelEmbedding}(\varphi) \]

\[ \gamma := \kappa \cdot \pi_1 \cdot \iota_1 \]
\[ \pi_i := \text{ProjectionInFactorOfDirectSum} \left( (M_1, M_2), i \right), \; i = 1, 2 \]

\[
\begin{align*}
\pi_1 & := \text{ProjectionInFactorOfDirectSum} \left( [ M_1, M_2 ], 1 \right); \\
\pi_2 & := \text{ProjectionInFactorOfDirectSum} \left( [ M_1, M_2 ], 2 \right);
\end{align*}
\]

\[ \varphi := \pi_1 \cdot \iota_1 - \pi_2 \cdot \iota_2 \]

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

\[ \gamma := \kappa \cdot \pi_1 \cdot \iota_1 \]
\[ \pi_i \ := \ \text{ProjectionInFactorOfDirectSum} \left( \left( M_1, M_2 \right), i \right), \ i = 1, 2 \]

\begin{verbatim}
  pil := ProjectionInFactorOfDirectSum( [ M1, M2 ], 1 );
  pi2 := ProjectionInFactorOfDirectSum( [ M1, M2 ], 2 );
\end{verbatim}

\[ \varphi \ := \ \pi_1 \cdot \iota_1 - \pi_2 \cdot \iota_2 \]

\begin{verbatim}
  lambda := PreCompose( pil, iota1 );
  phi := lambda - PreCompose( pi2, iota2 );
\end{verbatim}

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

\[ \gamma \ := \ \kappa \cdot \pi_1 \cdot \iota_1 \]
\[ \pi_i := \text{ProjectionInFactorOfDirectSum} \left( (M_1, M_2), i \right), \ i = 1, 2 \]

\[
\begin{align*}
\pi_1 &= \text{ProjectionInFactorOfDirectSum}( [ M_1, M_2 ], 1 ); \\
\pi_2 &= \text{ProjectionInFactorOfDirectSum}( [ M_1, M_2 ], 2 ); \\
\phi &= \pi_1 \cdot \iota_1 - \pi_2 \cdot \iota_2 \\
\lambda &= \text{PreCompose}( \pi_1, \iota_1 ); \\
\phi &= \lambda - \text{PreCompose}( \pi_2, \iota_2 ); \\
\kappa &= \text{KernelEmbedding}( \phi ) \\
\kappa &= \text{KernelEmbedding}( \phi ) \\
\gamma &= \kappa \cdot \pi_1 \cdot \iota_1 \\
\end{align*}
\]
\( \pi_i := \text{ProjectionInFactorOfDirectSum}((M_1, M_2), i), \ i = 1, 2 \)

\[
\begin{align*}
\pi_1 & := \text{ProjectionInFactorOfDirectSum}( [ M_1, M_2 ], 1 ); \\
\pi_2 & := \text{ProjectionInFactorOfDirectSum}( [ M_1, M_2 ], 2 ); \\
\end{align*}
\]

\( \varphi := \pi_1 \cdot \iota_1 - \pi_2 \cdot \iota_2 \)

\[
\begin{align*}
\lambda & := \text{PreCompose}( \pi_1, \iota_1 ); \\
\phi & := \lambda - \text{PreCompose}( \pi_2, \iota_2 ); \\
\end{align*}
\]

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

\[
\begin{align*}
\kappa & := \text{KernelEmbedding}( \phi ); \\
\end{align*}
\]

\( \gamma := \kappa \cdot \pi_1 \cdot \iota_1 \)

\[
\begin{align*}
\gamma & := \text{PreCompose}( \kappa, \lambda ); \\
\end{align*}
\]
\[
\pi_1 := \text{ProjectionInFactorOfDirectSum}( \{ M_1, M_2 \}, 1 ); \\
\pi_2 := \text{ProjectionInFactorOfDirectSum}( \{ M_1, M_2 \}, 2 ); \\
\phi := \pi_1 \cdot \iota_1 - \pi_2 \cdot \iota_2; \\
\lambda := \text{PreCompose}( \pi_1, \iota_1 ); \\
\phi := \lambda - \text{PreCompose}( \pi_2, \iota_2 ); \\
\kappa := \text{KernelEmbedding}( \phi ); \\
\gamma := \text{PreCompose}( \kappa, \lambda );
\]
Translation to CAP

```plaintext
pi1 := ProjectionInFactorOfDirectSum( [ M1, M2 ], 1 );
pi2 := ProjectionInFactorOfDirectSum( [ M1, M2 ], 2 );
lambda := PreCompose( pi1, iota1 );
phi := lambda - PreCompose( pi2, iota2 );
kappa := KernelEmbedding( phi );
gamma := PreCompose( kappa, lambda );
```
IntersectionOfSubobject := 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 := PreCompose( pi1, iota1 );
phi := lambda - PreCompose( pi2, iota2 );
kappa := KernelEmbedding( phi );
gamma := PreCompose( kappa, lambda );
return gamma;
end;
IntersectionOfSubobject := function( iota1, iota2 )

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

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

lambda := PreCompose( pi1, iota1 );
phi := lambda - PreCompose( pi2, iota2 );
kappa := KernelEmbedding( phi );
gamma := PreCompose( kappa, lambda );

return gamma;
IntersectionOfSubobject := function( iota1, iota2 )

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

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

lambda := PreCompose( pi1, iota1 );
phi := lambda - PreCompose( pi2, iota2 );
kappa := KernelEmbedding( phi );
gamma := PreCompose( kappa, lambda );

return gamma;
end;
IntersectionOfSubobject := 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 := PreCompose( pi1, iota1 );
    phi := lambda - PreCompose( pi2, iota2 );
    kappa := KernelEmbedding( phi );
    gamma := PreCompose( kappa, lambda );
    return gamma;
end;
Computing the intersection: \( \mathbb{Q}\text{-vec} \)

Compute the intersection of

\[
\begin{align*}
\nu_1 &:= \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \\
M_1 &\to N \\
2 &\parallel 3
\end{align*}
\]

\[
\begin{align*}
\nu_2 &:= \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \\
N &\to M_2 \\
3 &\parallel 2
\end{align*}
\]
Computing the intersection: $\mathbb{Q}$-vec

Compute the intersection of

$$\nu_1 := \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

$$\nu_2 := \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

\[
\begin{array}{c}
M_1 \xrightarrow{\|} 2 \\
\| \\
N \xleftarrow{\|} 3 \\
\| \\
M_2 \xrightarrow{\|} 2
\end{array}
\]

\[
\begin{array}{c}
\nu_1 := \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \\
\nu_2 := \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}
\end{array}
\]

gap> gamma := IntersectionOfSubobject( iota1, iota2 );

<A morphism in the category of matrices over $\mathbb{Q}$>
Computing the intersection: $\mathbb{Q}$-vec

Compute the intersection of

\[ M_1 \oplus 2 \rightarrow N \leftarrow M_2 \oplus 2 \]

\[ \nu_1 := \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \]
\[ \nu_2 := \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \]

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

\text{gap> Display( gamma );}
\text{[ [ 1, 1, 0 ] ]}

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

Homomorphism structures and applications
$R$ is a ring with identity, not necessarily commutative.
- $R$ is a ring with identity, not necessarily commutative.
- $R$ is supposed to be \textbf{computable}, that is, we have algorithms for
Conventions

- \( R \) is a ring with identity, not necessarily commutative.
- \( R \) is supposed to be \textbf{computable}, that is, we have algorithms for
  \( \in, =, 0, 1, +, -, \cdot, \)
Conventions

- $R$ is a ring with identity, not necessarily commutative.
- $R$ is supposed to be **computable**, that is, we have algorithms for
  1. $\in$, $=$, 0, 1, $+$, $-$, $\cdot$,
  2. solving one-sided linear systems of matrices with entries in $R$. 
Conventions

- $R$ is a ring with identity, not necessarily commutative.
- $R$ is supposed to be **computable**, that is, we have algorithms for
  1. $\in$, $=, 0, 1, +, -, \cdot$,
  2. solving one-sided linear systems of matrices with entries in $R$.
- $\text{Rows}_R$ is the category of matrices over $R$. 
1 Motivation

2 Homomorphism structures

3 Applications
Main aim

- Compute lifts in $\mathsf{fpres}_R$
Main aim

- Compute lifts in $\text{fpres}_R$
- That is, for given $\alpha$ and $\beta$ find $\xi$ in the following diagram:

$$
\begin{array}{c}
P \\
\downarrow^{\alpha} \\
M \xrightarrow{\beta} N \\
\uparrow_{\xi}
\end{array}
$$

That is, solve the one-sided linear system $\xi \cdot \beta = \alpha$. 

Posur, Saleh, Zickgraf (Siegen)
- Compute lifts in $\text{fpres}_R$
- That is, for given $\alpha$ and $\beta$ find $\xi$ in the following diagram:

\[
\begin{array}{ccc}
M & \xrightarrow{\beta} & N \\
\downarrow{\alpha} & & \\
\uparrow{\xi} \\
P
\end{array}
\]

- That is, solve the one-sided linear system

\[
\xi \cdot \beta = \alpha
\]

for $\xi$. 

Posur, Saleh, Zickgraf (Siegen)
Solving a **one-sided** linear system in $\text{fpres}_R$
Solving a **one-sided** linear system in \( \text{f}\text{p}\text{r}\text{e}s_R \) means solving a **two-sided** linear system of matrices with entries in \( R \).
Solving a **one-sided** linear system in $\text{fpres}_R$ means solving a **two-sided** linear system of matrices with entries in $R$. Why?
Solving a **one-sided** linear system in $\mathfrak{f}_{\mathfrak{p}_{\mathfrak{r}}}$ means solving a **two-sided** linear system of matrices with entries in $\mathbb{R}$. Why?

Consider the data structure of $\mathfrak{f}_{\mathfrak{p}_{\mathfrak{r}}}$:
Solving a **one-sided** linear system in $\text{fpres}_R$ means solving a **two-sided** linear system of matrices with entries in $R$. Why?

Consider the data structure of $\text{fpres}_R$:

$$
\begin{array}{c}
P \\
\downarrow \alpha \\
M \\
\downarrow \beta \\
N
\end{array}
$$

$X$ occurs both on the left and on the right.
Solving a **one-sided** linear system in $\text{fpres}_R$ means solving a **two-sided** linear system of matrices with entries in $R$. Why?

Consider the data structure of $\text{fpres}_R$:

\[
P \quad \xrightarrow{\xi} \quad M \quad \xleftarrow{\beta} \quad N \quad \xrightarrow{\alpha} \quad \text{X} \quad \xrightarrow{} \quad \text{X}' \quad \xrightarrow{} \quad \text{X}''
\]
Solving a **one-sided** linear system in $\text{f} \text{pres}_R$ means solving a **two-sided** linear system of matrices with entries in $R$. Why?

Consider the data structure of $\text{f} \text{pres}_R$:

\[
\begin{align*}
P & \xrightarrow{\alpha} M \xleftarrow{\xi} N \\
R^{1 \times p} & \xrightarrow{\langle P \rangle} R^{1 \times m} & R^{1 \times n} & \xleftarrow{B} R^{1 \times m} \\
R^{1 \times p} & \xleftarrow{\langle P \rangle} R^{1 \times m} & R^{1 \times n} & \xrightarrow{B} R^{1 \times m}
\end{align*}
\]
Motivation

Problem

- Solving a **one-sided** linear system in $\mathsf{fpres}_R$ means solving a **two-sided** linear system of matrices with entries in $R$. Why?

  Consider the data structure of $\mathsf{fpres}_R$:

  We have to find $X$ such that
Solving a **one-sided** linear system in $\text{fpres}_R$ means solving a **two-sided** linear system of matrices with entries in $R$. Why?

Consider the data structure of $\text{fpres}_R$:

We have to find $X$ such that

- $\xi$ is well-defined:
Solving a **one-sided** linear system in $\text{fpres}_R$ means solving a **two-sided** linear system of matrices with entries in $R$. Why?

Consider the data structure of $\text{fpres}_R$:

$$
\begin{align*}
\begin{array}{cccc}
\rightarrow & P & \downarrow & \alpha \\
\downarrow & \beta & \rightarrow & N \\
\leftarrow & M & \uparrow & \xi
\end{array}
\end{align*}
$$

We have to find $X$ such that

- $\xi$ is well-defined: $P \cdot X = X' \cdot M$ for some $X'$
Solving a **one-sided** linear system in $\text{fpres}_R$ means solving a **two-sided** linear system of matrices with entries in $R$. Why?

Consider the data structure of $\text{fpres}_R$:

\[ P \quad M \quad N \]

\[ \xi \quad \alpha \quad \beta \]

We have to find $X$ such that

1. $\xi$ is well-defined: $P \cdot X = X' \cdot M$ for some $X'$
2. $\xi \cdot \beta = \alpha$:
Solving a **one-sided** linear system in $\mathsf{fpres}_R$ means solving a **two-sided** linear system of matrices with entries in $R$. Why?

Consider the data structure of $\mathsf{fpres}_R$:

$$
\begin{array}{c}
P \\
\downarrow \alpha \\
M \xrightarrow{\beta} N \\
\uparrow \xi
\end{array}
\quad
\begin{array}{c}
\mathsf{R}^{1 \times p} \\
\downarrow \langle P \rangle \\
\mathsf{R}^{1 \times m} \\
\downarrow \langle M \rangle
\end{array}
\quad
\begin{array}{c}
X \\
\uparrow \beta
\end{array}
\quad
\begin{array}{c}
\mathsf{R}^{1 \times n} \\
\downarrow A \\
\mathsf{B}
\end{array}
$$

We have to find $X$ such that

1. $\xi$ is well-defined: $P \cdot X = X' \cdot M$ for some $X'$
2. $\xi \cdot \beta = \alpha$: $X \cdot B = A$
Solving a one-sided linear system in $\text{fpres}_R$ means solving a two-sided linear system of matrices with entries in $R$. Why?

Consider the data structure of $\text{fpres}_R$:

We have to find $X$ such that

1. $\xi$ is well-defined: $P \cdot X = X' \cdot M$ for some $X'$
2. $\xi \cdot \beta = \alpha$: $X \cdot B = A + X'' \cdot N$ for some $X''$
Solving a **one-sided** linear system in $\text{fpres}_R$ means solving a **two-sided** linear system of matrices with entries in $R$. Why?

Consider the data structure of $\text{fpres}_R$:

![Diagram]

We have to find $X$ such that

1. $\xi$ is well-defined: $P \cdot X = X' \cdot M$ for some $X'$
2. $\xi \cdot \beta = \alpha$: $X \cdot B = A + X'' \cdot N$ for some $X''$

$X$ occurs both on the left and on the right.
Solution for commutative rings

For example, consider $R = \mathbb{Q}$. 
For example, consider $R = \mathbb{Q}$.

Define $\text{vec}: \mathbb{Q}^{n \times m} \rightarrow \mathbb{Q}^{1 \times nm}$
For example, consider $R = \mathbb{Q}$.

Define $\text{vec}: \mathbb{Q}^{n \times m} \rightarrow \mathbb{Q}^{1 \times nm}$ which concatenates the rows of a matrix $M$ to get a long row vector.
Solution for commutative rings

For example, consider $R = \mathbb{Q}$.

Define $\text{vec}: \mathbb{Q}^{n \times m} \rightarrow \mathbb{Q}^{1 \times nm}$ which concatenates the rows of a matrix $M$ to get a long row vector.

Then $\text{vec}(M \cdot X \cdot N) = \text{vec}(X) \cdot (M^T \otimes N)$
For example, consider $R = \mathbb{Q}$.

Define $\text{vec}: \mathbb{Q}^{n \times m} \to \mathbb{Q}^{1 \times nm}$ which concatenates the rows of a matrix $M$ to get a long row vector.

Then $\text{vec}(M \cdot X \cdot N) = \text{vec}(X) \cdot (M^T \otimes N)$

This explicitly depends on the commutativity of the ring (obvious for $1 \times 1$ matrices).
Solution for commutative rings

Let $S$ be a commutative ring.

Define $\text{vec} : \mathbb{Q}^{n \times m} \to \mathbb{Q}^{1 \times nm}$ which concatenates the rows of a matrix $M$ to get a long row vector.

Then $\text{vec}(M \cdot X \cdot N) = \text{vec}(X) \cdot (M^T \otimes N)$

This explicitly depends on the commutativity of the ring (obvious for $1 \times 1$ matrices).
Motivation

Solution for commutative rings

- Let $S$ be a commutative ring.
- Define $\text{vec}: S^{n \times m} \rightarrow S^{1 \times nm}$ which concatenates the rows of a matrix $M$ to get a long row vector.
- Then $\text{vec}(M \cdot X \cdot N) = \text{vec}(X) \cdot (M^T \otimes N)$
- This explicitly depends on the commutativity of the ring (obvious for $1 \times 1$ matrices).
Let $S$ be a commutative ring.

Define $\text{vec} : S^{n \times m} \rightarrow S^{1 \times nm}$ which concatenates the rows of a matrix $M$ to get a long row vector.

Then $\text{vec}(M \cdot X \cdot N) = \text{vec}(X) \cdot (M^T \otimes N)$

This explicitly depends on the commutativity of the ring (obvious for $1 \times 1$ matrices).

**Aim**

Find a category theoretical abstraction of this trick.
1 Motivation

2 Homomorphism structures

3 Applications
Definition

Let $\mathcal{C}$ and $\mathcal{D}$ be categories.
Definition

Let $\mathcal{C}$ and $\mathcal{D}$ be categories. A $\mathcal{D}$-homomorphism structure for $\mathcal{C}$ consists of the following data:

- A distinguished object $1 \in \mathcal{D}$
- A functor $H : \mathcal{C}^{\text{op}} \times \mathcal{C} \to \mathcal{D}$
- An isomorphism $\nu : \text{Hom}_\mathcal{C}(A, B) \cong \text{Hom}_\mathcal{D}(1, H(A, B))$ natural in $A, B \in \mathcal{C}$

Moreover, if we are in the context of Ab-categories, we require $H$ to be bilinear.
Definition

Let \( \mathcal{C} \) and \( \mathcal{D} \) be categories. A \( \mathcal{D} \)-homomorphism structure for \( \mathcal{C} \) consists of the following data:

- A distinguished object \( 1 \in \mathcal{D} \)
Definition

Let \( \mathcal{C} \) and \( \mathcal{D} \) be categories. A \( \mathcal{D} \)-homomorphism structure for \( \mathcal{C} \) consists of the following data:

- A distinguished object \( 1 \in \mathcal{D} \)
- A functor \( H: \mathcal{C}^{\text{op}} \times \mathcal{C} \to \mathcal{D} \)

Moreover, if we are in the context of Ab-categories, we require \( H \) to be bilinear.
Definition

Let $\mathcal{C}$ and $\mathcal{D}$ be categories. A $\mathcal{D}$-homomorphism structure for $\mathcal{C}$ consists of the following data:

- A distinguished object $1 \in \mathcal{D}$
- A functor $H: \mathcal{C}^{\text{op}} \times \mathcal{C} \to \mathcal{D}$
- An isomorphism $\nu: \text{Hom}_\mathcal{C}(A, B) \cong \text{Hom}_\mathcal{D}(1, H(A, B))$ natural in $A, B \in \mathcal{C}$
Definition

Let $\mathcal{C}$ and $\mathcal{D}$ be categories. A $\mathcal{D}$-homomorphism structure for $\mathcal{C}$ consists of the following data:

- A distinguished object $1 \in \mathcal{D}$
- A functor $H: \mathcal{C}^{\text{op}} \times \mathcal{C} \to \mathcal{D}$
- An isomorphism $\nu: \text{Hom}_\mathcal{C}(A, B) \cong \text{Hom}_\mathcal{D}(1, H(A, B))$ natural in $A, B \in \mathcal{C}$

Moreover, if we are in the context of Ab-categories, we require $H$ to be bilinear.
Homomorphism structures

Unwrapping the definition

This implies that the following diagram is commutative for all possible choices of $\alpha$, $\beta$, and $\xi$:

$$
\nu(\xi) \cdot \alpha \cdot \xi \cdot \beta = \nu(\xi) \cdot H(\alpha, \beta)
$$

$$
\nu \circ \text{Hom}_C(\alpha, \beta) \circ \text{Hom}_D(id_1, H(\alpha, \beta))
$$
Third property

An isomorphism $\nu : \text{Hom}_C(A, B) \cong \text{Hom}_D(1, H(A, B))$ natural in $A, B \in C$
An isomorphism \( \nu : \text{Hom}_C(A, B) \cong \text{Hom}_D(1, H(A, B)) \) natural in \( A, B \in C \)

This implies that the following diagram is commutative for all possible choices of \( \alpha, \beta \) and \( \xi \):
An isomorphism \( \nu : \text{Hom}_C(A, B) \cong \text{Hom}_D(1, H(A, B)) \) natural in \( A, B \in C \)

This implies that the following diagram is commutative for all possible choices of \( \alpha, \beta \) and \( \xi \):

\[
\begin{array}{ccc}
\xi & \overset{\nu}{\longrightarrow} & \nu(\xi) \\
\downarrow_{\text{Hom}_C(\alpha, \beta)} & & \downarrow_{\text{Hom}_D(\text{id}_1, H(\alpha, \beta))} \\
\alpha \cdot \xi \cdot \beta & \overset{\nu}{\longrightarrow} & \nu(\alpha \cdot \xi \cdot \beta) = \nu(\xi) \cdot H(\alpha, \beta)
\end{array}
\]
Let $S$ be a commutative ring.
Example

- Let $S$ be a commutative ring.
- $\mathcal{C} = \mathcal{D} = \text{Rows}_S$
Example

1. Let $S$ be a commutative ring.
2. $\mathcal{C} = \mathcal{D} = \text{Rows}_S$
3. $H$ on morphisms: $H(M, N) = M^T \otimes N$
Example

- Let $S$ be a commutative ring.
- $C = D = \text{Rows}_S$
- $H$ on morphisms: $H(M, N) = M^T \otimes N$
- $H$ on objects: $H(S^1 \times n, S^1 \times m) = S^1 \times n \cdot m$
Example

- Let $S$ be a commutative ring.
- $\mathcal{C} = \mathcal{D} = \text{Rows}_S$
- $H$ on morphisms: $H(M, N) = M^T \otimes N$
- $H$ on objects: $H(S^1 \times n, S^1 \times m) = S^1 \times n \cdot m$
- $\nu(M) = \text{vec}(M)$
Example

- Let $S$ be a commutative ring.
- $\mathcal{C} = \mathcal{D} = \text{Rows}_S$
- $H$ on morphisms: $H(M, N) = M^T \otimes N$
- $H$ on objects: $H(S^1 \times n, S^1 \times m) = S^1 \times n \cdot m$
- $\nu(M) = \text{vec}(M)$
- $1 = S^1 \times 1$
Since $H$ is bilinear, it is already determined by its values on $S^{1 \times 1}$ and $\text{Hom}_{\text{Rows}}(S^{1 \times 1}, S^{1 \times 1})$: 
Since $H$ is bilinear, it is already determined by its values on $S^1 \times 1$ and $\text{Hom}_{\text{Rows}_S}(S^1 \times 1, S^1 \times 1)$:

$$(S^1 \times 1, S^1 \times 1) \mapsto S^1 \times 1$$

Note: in this sense, $H$ can be interpreted as the (enriched) $\text{Hom}$-functor on $\text{Rows}_S$. Similarly, $\nu$ is determined by its values on $\text{Hom}_{\text{Rows}_S}(S^1 \times 1, S^1 \times 1)$:

$$(c) \mapsto (c)$$

Naturality of $\nu$:

$$(a \cdot c \cdot b) = (c) \cdot (a \cdot b)$$
Since $H$ is bilinear, it is already determined by its values on $S^1 \times 1$ and $\text{Hom}_{\text{RowSS}}(S^1 \times 1, S^1 \times 1)$:
- $(S^1 \times 1, S^1 \times 1) \mapsto S^1 \times 1$
- $((a), (b)) \mapsto (a \cdot b)$
Since $H$ is bilinear, it is already determined by its values on $S^1 \times 1$ and $\text{Hom}_{\text{Rows}_S}(S^1 \times 1, S^1 \times 1)$:

- $(S^1 \times 1, S^1 \times 1) \mapsto S^1 \times 1$
- $((a), (b)) \mapsto (a \cdot b)$

Note: in this sense, $H$ can be interpreted as the (enriched) $\text{Hom}$-functor on $\text{Rows}_S$. 
Since $H$ is bilinear, it is already determined by its values on $S^1 \times 1$ and $\text{Hom}_{\text{Rows}_S}(S^1 \times 1, S^1 \times 1)$:

- $(S^1 \times 1, S^1 \times 1) \mapsto S^1 \times 1$
- $((a), (b)) \mapsto (a \cdot b)$

Note: in this sense, $H$ can be interpreted as the (enriched) Hom-functor on $\text{Rows}_S$.

Similarly, $\nu$ is determined by its values on $\text{Hom}_{\text{Rows}_S}(S^1 \times 1, S^1 \times 1)$:
Since $H$ is bilinear, it is already determined by its values on $S^1 \times 1$ and $\text{Hom}_{\text{Rows}_S}(S^1 \times 1, S^1 \times 1)$:

- $(S^1 \times 1, S^1 \times 1) \mapsto S^1 \times 1$
- $((a), (b)) \mapsto (a \cdot b)$

Note: in this sense, $H$ can be interpreted as the (enriched) $\text{Hom}$-functor on $\text{Rows}_S$.

Similarly, $\nu$ is determined by its values on $\text{Hom}_{\text{Rows}_S}(S^1 \times 1, S^1 \times 1)$:

$(c) \mapsto (c)$
Making use of the bilinearity

Since $H$ is bilinear, it is already determined by its values on $S^1 \times 1$ and $\text{Hom}_{\text{Rows}_S}(S^1 \times 1, S^1 \times 1)$:
- $(S^1 \times 1, S^1 \times 1) \mapsto S^1 \times 1$
- $((a), (b)) \mapsto (a \cdot b)$

Note: in this sense, $H$ can be interpreted as the (enriched) Hom-functor on $\text{Rows}_S$.

Similarly, $\nu$ is determined by its values on $\text{Hom}_{\text{Rows}_S}(S^1 \times 1, S^1 \times 1)$:
- $(c) \mapsto (c)$

Naturality of $\nu$: $(a \cdot c \cdot b) = (c) \cdot (a \cdot b)$
Problem: for a non-commutative ring and $H$ and $\nu$ as above, we lose the naturality of $\nu$. 

Solution: assume that we can find a subring $S$ of the center of $R$, such that $R$ is finitely presented as an $S$-module. That is, $S \cdot R \sim S \cdot 1 \times m \cdot \langle M \rangle$. Then multiplication from left and right with elements of $R$ is $S$-linear.
Non-commutative example

Problem: for a non-commutative ring and $H$ and $\nu$ as above, we lose the naturality of $\nu$.

Solution: assume that we can find a subring $S$ of the center of $R$, such that $R$ is finitely presented as an $S$-module.
Problem: for a non-commutative ring and $H$ and $\nu$ as above, we lose the naturality of $\nu$.

Solution: assume that we can find a subring $S$ of the center of $R$, such that $R$ is finitely presented as an $S$-module.

That is, $S R \cong \frac{S^1 \times M}{\langle M \rangle}$. 
Non-commutative example

- Problem: for a non-commutative ring and $H$ and $\nu$ as above, we lose the naturality of $\nu$.
- Solution: assume that we can find a subring $S$ of the center of $R$, such that $R$ is finitely presented as an $S$-module.
- That is, $sR \cong \frac{S^1 \times m}{\langle M \rangle}$.
- Then multiplication from left and right with elements of $R$ is $S$-linear.
Using this, we can find an $\text{fpres}_S$-homomorphism structure $(H, 1, \nu)$ for $\text{Rows}_R$.
Using this, we can find an $\text{fpres}_S$-homomorphism structure $(H, 1, \nu)$ for $\text{Rows}_R$:

$H : \text{Rows}_R^{\text{op}} \times \text{Rows}_R \rightarrow \text{fpres}_S$:
Using this, we can find an \( \text{fpres}_S \)-homomorphism structure \((H, 1, \nu)\) for \(\text{Rows}_R\):

\[
H: \text{Rows}_R^{\text{op}} \times \text{Rows}_R \to \text{fpres}_S:
\]

\[
(R^{1 \times 1}, R^{1 \times 1}) \mapsto \frac{S^{1 \times m}}{\langle M \rangle}
\]
Using this, we can find an $\text{fpres}_S$-homomorphism structure $(H, 1, \nu)$ for $\text{Rows}_R$:

$H : \text{Rows}_R^{\text{op}} \times \text{Rows}_R \rightarrow \text{fpres}_S$:

- $(R^{1 \times 1}, R^{1 \times 1}) \mapsto \frac{S^{1 \times m}}{\langle M \rangle}$
- $((a), (b)) \mapsto a \cdot - \cdot b$ expressed as a matrix in $S^{m \times m}$
Using this, we can find an $\text{fpres}_S$-homomorphism structure $(H, 1, \nu)$ for $\text{Rows}_R$:

$H : \text{Rows}_R^{\text{op}} \times \text{Rows}_R \rightarrow \text{fpres}_S$:

- $(R^1 \times 1, R^1 \times 1) \mapsto S^{1 \times m}/\langle M \rangle$
- $((a), (b)) \mapsto a \cdot - \cdot b$ expressed as a matrix in $S^{m \times m}$

$\nu$ is essentially the isomorphism $\varphi : sR \rightarrow S^{1 \times m}/\langle M \rangle$
Using this, we can find an \( \text{fpres}_S \)-homomorphism structure \((H, 1, \nu)\) for \(\text{Rows}_R\):

\[
H : \text{Rows}_R^{\text{op}} \times \text{Rows}_R \rightarrow \text{fpres}_S : \\
(R^1 \times 1, R^1 \times 1) \mapsto S^{1 \times m}_{\langle M \rangle} \\
((a), (b)) \mapsto a \cdot - \cdot b \text{ expressed as a matrix in } S^{m \times m}
\]

\(\nu\) is essentially the isomorphism \(\varphi : sR \rightarrow S^{1 \times m}_{\langle M \rangle}\)

Naturality of \(\nu\): \(\varphi(a \cdot c \cdot b) = \varphi(c) \cdot (a \cdot - \cdot b)\)
As above, this data uniquely defines the homomorphism structure for all of $\text{Rows}_R$. 
As above, this data uniquely defines the homomorphism structure for all of \( \text{Rows}_R \).

In particular this works for the exterior algebra \( E \) of \( \mathbb{Q}^n \) over its center or over \( \mathbb{Q} \).
1 Motivation

2 Homomorphism structures

3 Applications
Let $\mathcal{C}$ and $\mathcal{D}$ be additive categories.
Let $\mathcal{C}$ and $\mathcal{D}$ be additive categories.

A linear system in $\mathcal{C}$ is a collection of morphisms $\alpha_{ij}$, $\beta_{ij}$ and $\gamma_i$ of the following form:

\[
\alpha_{11} \cdot X_1 \cdot \beta_{11} + \ldots + \alpha_{1n} \cdot X_n \cdot \beta_{1n} = \gamma_1
\]
\[
\vdots
\]
\[
\alpha_{m1} \cdot X_1 \cdot \beta_{m1} + \ldots + \alpha_{mn} \cdot X_n \cdot \beta_{mn} = \gamma_m
\]

where the $X_j$ are unknown morphisms.
Assume we have a $\mathcal{D}$-homomorphism structure for $\mathcal{C}$. 
Assume we have a $\mathcal{D}$-homomorphism structure for $\mathcal{C}$.

Using that we can transfer a two-sided linear system in $\mathcal{C}$ to a one-sided linear system in $\mathcal{D}$ by using $\nu(\alpha \cdot \xi \cdot \beta) = \nu(\xi) \cdot H(\alpha, \beta)$:
Two-sided linear systems (2)

- Assume we have a $\mathcal{D}$-homomorphism structure for $\mathcal{C}$.
- Using that we can transfer a two-sided linear system in $\mathcal{C}$ to a one-sided linear system in $\mathcal{D}$ by using $\nu(\alpha \cdot \xi \cdot \beta) = \nu(\xi) \cdot H(\alpha, \beta)$:

$$
\begin{array}{c}
\oplus_i H(A_i, D_i) \\
\downarrow (\nu(\gamma_i))_i \\
\oplus_j H(B_j, C_j)
\end{array}
\xrightarrow{(\nu(X_j))_j}
\begin{array}{c}
\oplus_i H(A_i, D_i) \\
\downarrow (H(\alpha_{ij}, \beta_{ij}))_{ji}
\end{array}
\xleftarrow{(\nu(X_j))_j}
\begin{array}{c}
\oplus_j H(B_j, C_j)
\end{array}
$$
Lifts in $\text{fpres}_R$

$\text{fpres}_R$

lift, i.e. one-sided
Lifts in $\text{fpres}_R$

- $\text{fpres}_R$ lift, i.e. one-sided

- $\text{Rows}_R$ two-sided
Lifts in $\text{fpres}_R$

- $\text{fpres}_R$ lift, i.e. one-sided
- $\text{Rows}_R$ two-sided
- $\text{fpres}_S$ one-sided
Lifts in $\text{fpres}_R$

- Lift, i.e. one-sided
- $\text{fpres}_R$
- $\text{Rows}_R$
  - Two-sided
- $\text{fpres}_S$
  - One-sided
  - $\text{Rows}_S$
    - Two-sided
Lifts in $\text{fpres}_R$

- $\text{fpres}_R$ lift, i.e. one-sided
- $\text{Rows}_R$ two-sided
- $\text{fpres}_S$ one-sided
- $\text{Rows}_S$ two-sided
- $\text{Rows}_S$ one-sided
Lifts in $\text{fpres}_R$

fpres$_R$

lift, i.e. one-sided

$\text{Rows}_R$

two-sided

fpres$_S$

one-sided

$\text{Rows}_S$

one-sided
CAP basic operations

- HomomorphismStructureOnObjects
CAP basic operations

- HomomorphismStructureOnObjects
- HomomorphismStructureOnMorphisms
CAP basic operations

- HomomorphismStructureOnObjects
- HomomorphismStructureOnMorphisms
- DistinguishedObjectOfHomomorphismStructure
CAP basic operations

- HomomorphismStructureOnObjects
- HomomorphismStructureOnMorphisms
- DistinguishedObjectOfHomomorphismStructure
- InterpretMorphismAsMorphismFromDistinguishedObjectToHomomorphismStructure
CAP basic operations

- HomomorphismStructureOnObjects
- HomomorphismStructureOnMorphisms
- DistinguishedObjectOfHomomorphismStructure
- InterpretMorphismAsMorphismFromDistinguishedObjectToHomomorphismStructure
- InterpretMorphismFromDistinguishedObjectToHomomorphismStructureAsMorphism
CAP basic operations

- HomomorphismStructureOnObjects
- HomomorphismStructureOnMorphisms
- DistinguishedObjectOfHomomorphismStructure
- InterpretMorphismAsMorphismFromDistinguishedObjectToHomomorphismStructure
- InterpretMorphismFromDistinguishedObjectToHomomorphismStructureAsMorphism

⇒ SolveLinearSystemInAbCategory
Part III

Stable categories and homotopy categories in CAP
Stabilisation of a category by its projective objects

**Motto**: We want to identify projective objects with the zero object.

- **Motto**: We want to identify projective objects with the zero object.
**Motto:** We want to identify projective objects with the zero object. Hence, morphisms that factor through a projective object should be treated as zero morphisms, and ...
Motto: We want to identify projective objects with the zero object. Hence, morphisms that factor through a projective object should be treated as zero morphisms, and . . . any two morphisms whose subtraction factors through a projective object should be treated as identical morphisms.
Motto: We want to identify projective objects with the zero object. Hence, morphisms that factor through a projective object should be treated as zero morphisms, and . . . any two morphisms whose subtraction factors through a projective object should be treated as identical morphisms.

Definition
Let $R$ be a ring and $R$-mod its module category. We define the stable module category of $R$, denoted by $R\text{-mod}$ as follows:
Stabilisation of a category by its projective objects

**Motto:** We want to identify projective objects with the zero object. Hence, morphisms that factor through a projective object should be treated as zero morphisms, and . . . any two morphisms whose subtraction factors through a projective object should be treated as identical morphisms.

**Definition**

Let \( R \) be a ring and \( R\text{-mod} \) its module category. We define the stable module category of \( R \), denoted by \( R\text{-mod} \) as follows:

1. \( \text{Obj}(R\text{-mod}) := \text{Obj}(R\text{-mod}) \),
Stabilisation of a category by its projective objects

- **Motto**: We want to identify projective objects with the zero object. Hence, morphisms that factor through a projective object should be treated as zero morphisms, and ... any two morphisms whose subtraction factors through a projective object should be treated as identical morphisms.

**Definition**

Let $R$ be a ring and $R$-mod its module category. We define the stable module category of $R$, denoted by $\text{R-mod}$ as follows:

1. $\text{Obj}(\text{R-mod}) := \text{Obj}(\text{R-mod})$,
2. For $a, b \in \text{R-mod}$, we define

\[
\text{Hom}_{\text{R-mod}}(a, b) := \text{Hom}_{\text{R-mod}}(a, b)/\sim,
\]

where $\varphi \sim \psi$ if $\varphi - \psi$ factors through a projective object.
Suppose $\varphi : a \rightarrow b$ is an $R$-homomorphism that factors through a projective object. Furthermore, let $\pi_b : P_b \twoheadrightarrow b$ be an epimorphism from some projective object $P_b$. Then
Suppose $\varphi : a \to b$ is an $R$-homomorphism that factors through a projective object. Furthermore, let $\pi_b : P_b \to b$ be an epimorphism from some projective object $P_b$. Then
Suppose $\varphi : a \to b$ is an $R$-homomorphism that factors through a projective object. Furthermore, let $\pi_b : P_b \to b$ be an epimorphism from some projective object $P_b$. Then

$P$ is projective object and the diagram commutes.
Suppose $\varphi : a \to b$ is an $R$-homomorphism that factors through a projective object. Furthermore, let $\pi_b : P_b \to b$ be an epimorphism from some projective object $P_b$. Then

\[
P \quad \xymatrix{ \exists \ar[dr] & a \ar[d] \ar[dl] \ar[dr] & \exists \\
\exists \ar[dr] & P \ar[d] \ar[dl] \ar[dr] \ar[d] & \exists \\
P_b \ar[r]_{\pi_b} & b \ar[dl]_{\varphi}}
\]

- $P$ is projective object and the diagram commutes.
Suppose $\varphi : a \to b$ is an $R$-homomorphism that factors through a projective object. Furthermore, let $\pi_b : P_b \to b$ be an epimorphism from some projective object $P_b$. Then

- $P$ is projective object and the diagram commutes.
- Because any morphism from a projective module is liftable to any epimorphism,
Suppose $\varphi : a \to b$ is an $R$-homomorphism that factors through a projective object. Furthermore, let $\pi_b : P_b \to b$ be an epimorphism from some projective object $P_b$. Then

- $P$ is projective object and the diagram commutes.
- Because any morphism from a projective module is liftable to any epimorphism, it is enough to decide liftability of $\varphi$ along $\pi_b$. 
A category $\mathcal{A}$ is called computable with enough projectives if $\mathcal{A}$ is computable and is equipped with following basic operations:

<table>
<thead>
<tr>
<th>Basic operation</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsProjective</td>
<td>$b$</td>
<td>$b$</td>
</tr>
<tr>
<td>SomeProjectiveObject</td>
<td>$P;b$</td>
<td></td>
</tr>
<tr>
<td>EpimorphismFrom</td>
<td>$\pi;b$</td>
<td>$b$</td>
</tr>
<tr>
<td>SomeProjectiveObject</td>
<td>$\alpha;P\mapsto b$</td>
<td>$\beta;a\mapsto b$</td>
</tr>
<tr>
<td>ProjectiveLift</td>
<td>$\delta;P\mapsto a$</td>
<td></td>
</tr>
</tbody>
</table>
A category $\mathcal{A}$ is called computable with enough projectives if $\mathcal{A}$ is computable and is equipped with the following basic operations:

<table>
<thead>
<tr>
<th>Basic operation</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsProjective</td>
<td>$b$</td>
<td>is $b$ projective?</td>
</tr>
</tbody>
</table>
A category $\mathcal{A}$ is called computable with enough projectives if $\mathcal{A}$ is computable and is equipped with the following basic operations:

<table>
<thead>
<tr>
<th>Basic operation</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsProjective</td>
<td>$b$</td>
<td>$\text{is } b \text{ projective?}$</td>
</tr>
<tr>
<td>SomeProjectiveObject</td>
<td>$b$</td>
<td>$P_b$</td>
</tr>
</tbody>
</table>
A category $\mathcal{A}$ is called computable with enough projectives if $\mathcal{A}$ is computable and is equipped with following basic operations:

<table>
<thead>
<tr>
<th>Basic operation</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsProjective</td>
<td>$b$</td>
<td>is $b$ projective?</td>
</tr>
<tr>
<td>SomeProjectiveObject</td>
<td>$b$</td>
<td>$P_b$</td>
</tr>
<tr>
<td>EpimorphismFrom-</td>
<td>$b$</td>
<td>$\pi_b : P_b \rightarrow b$</td>
</tr>
<tr>
<td>SomeProjectiveObject</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A category $\mathcal{A}$ is called computable with enough projectives if $\mathcal{A}$ is computable and is equipped with the following basic operations:

<table>
<thead>
<tr>
<th>Basic operation</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsProjective</td>
<td>$b$</td>
<td>is $b$ projective?</td>
</tr>
<tr>
<td>SomeProjectiveObject</td>
<td>$b$</td>
<td>$P_b$</td>
</tr>
<tr>
<td>EpimorphismFrom-SomeProjectiveObject</td>
<td>$b$</td>
<td>$\pi_b : P_b \to b$</td>
</tr>
<tr>
<td>ProjectiveLift</td>
<td>$(\alpha : P \to b, \beta : a \to b)$</td>
<td>$\delta : P \to a$</td>
</tr>
</tbody>
</table>
Back to our $R$-homomorphism $\varphi : a \to b$. 

The following code in GAP checks whether $\varphi$ factors through a projective:

```
gap> b := Range( phi );
gap> pi_b := EpimorphismFromSomeProjectiveObject( b );
gap> IsLiftable( phi, pi_b );
```
Back to our $R$-homomorphism $\varphi : a \to b$.

The following code in CAP checks whether $\varphi$ factors through a projective:

```gap
gap> b := Range( phi );
gap> pi_b := EpimorphismFromSomeProjectiveObject( b );
gap> IsLiftable( phi, pi_b );
```

Posur, Saleh, Zickgraf (Siegen)
Lifting objects as an abstraction of projective objects

Definition

Let $\mathcal{A}$ be an additive category. A system of lifting objects $\mathcal{L}_\mathcal{A}$ in $\mathcal{A}$ is a distinguished class of objects with the following properties:

1. For any object $a$ in $\mathcal{A}$, there exists a morphism $\ell_a : \mathcal{L}a \to a$, for some object $\mathcal{L}a \in \mathcal{L}_\mathcal{A}$.

2. For any morphism $\varphi : a \to b$, there is a lifting morphism $\mathcal{L}\varphi : \mathcal{L}a \to \mathcal{L}b$ such that the following diagram commutes:

$$
\begin{array}{ccc}
\mathcal{L}a & \xrightarrow{\ell_a} & a \\
\downarrow & & \downarrow \varphi \\
\mathcal{L}b & \xrightarrow{\mathcal{L}\varphi} & b
\end{array}
$$
Lifting objects as an abstraction of projective objects

Definition

Let $\mathcal{A}$ be an additive category. A system of lifting objects $\mathcal{L}_\mathcal{A}$ in $\mathcal{A}$ is a distinguished class of objects with the following properties:

1. For any object $a$ in $\mathcal{A}$, there exists a morphism $\ell_a : \mathcal{L}_a \to a$, for some object $\mathcal{L}_a \in \mathcal{L}_\mathcal{A}$.
Lifting objects as an abstraction of projective objects

Definition

Let $\mathcal{A}$ be an additive category. A system of lifting objects $\mathcal{L}_\mathcal{A}$ in $\mathcal{A}$ is a distinguished class of objects with the following properties:

1. For any object $a$ in $\mathcal{A}$, there exists a morphism $\ell_a : \mathcal{L}_a \to a$, for some object $\mathcal{L}_a \in \mathcal{L}_\mathcal{A}$.

2. For any morphism $\varphi : a \to b$, there is a lifting morphism $\mathcal{L}_\varphi : \mathcal{L}_a \to \mathcal{L}_b$ such that the following diagram commutes:

$$
\begin{array}{ccc}
\mathcal{L}_a & \xrightarrow{\ell_a} & a \\
\downarrow{\mathcal{L}_\varphi} & & \downarrow{\varphi} \\
\mathcal{L}_b & \xrightarrow{\ell_b} & b.
\end{array}
$$
Projective objects define a lifting system

For any additive category with enough projectives we have a lifting system defined by:

\[
\mathcal{L}_a := P_a \quad \ell_a := \pi_a \quad a
\]

\[
\mathcal{L}_b := P_b \quad \ell_b := \pi_b \quad b
\]

\[
\mathcal{L}_\varphi := \text{ProjectiveLift}(\pi_a \varphi, \pi_b)
\]

Diagram:

\[
\begin{array}{c}
\mathcal{L}_a := P_a \quad \ell_a := \pi_a \\
\mathcal{L}_b := P_b \quad \ell_b := \pi_b \\
\mathcal{L}_\varphi := \text{ProjectiveLift}(\pi_a \varphi, \pi_b)
\end{array}
\]
Stable categories by a lifting system

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

- If $\mathcal{L}_{\mathcal{A}}$ is a system of lifting objects for $\mathcal{A}$, then

$$\mathcal{I}_{\mathcal{L}_{\mathcal{A}}} = \{ \varphi : a \to b | \varphi \text{ factors through } \ell_b \}$$

is a two-sided ideal of morphisms in $\mathcal{A}$.
Let $\mathcal{A}$ be an additive category.

- If $\mathcal{L}_\mathcal{A}$ is a system of lifting objects for $\mathcal{A}$, then

$$\mathcal{I}_{\mathcal{L}_\mathcal{A}} = \{ \varphi : a \to b | \varphi \text{ factors through } \ell_b \}$$

is a two-sided ideal of morphisms in $\mathcal{A}$.

- We define the stable category of $\mathcal{A}$ by $\mathcal{L}_\mathcal{A}$ by

$$\text{Stab}_{\mathcal{L}_\mathcal{A}}(\mathcal{A}) := \mathcal{A}/\mathcal{I}_{\mathcal{L}_\mathcal{A}},$$
Let $\mathcal{A}$ be an additive category.

- If $\mathcal{L}_\mathcal{A}$ is a system of lifting objects for $\mathcal{A}$, then

$$\mathcal{I}_{\mathcal{L}_\mathcal{A}} = \{ \varphi : a \to b \mid \varphi \text{ factors through } \ell_b \}$$

is a two-sided ideal of morphisms in $\mathcal{A}$.

- We define the stable category of $\mathcal{A}$ by $\mathcal{L}_\mathcal{A}$ by

$$\text{Stab}_{\mathcal{L}_\mathcal{A}}(\mathcal{A}) := \mathcal{A}/\mathcal{I}_{\mathcal{L}_\mathcal{A}},$$

i.e.,

1. The object class is same as that of $\mathcal{A}$. 


Stable categories by a lifting system

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

- If $\mathcal{L}_\mathcal{A}$ is a system of lifting objects for $\mathcal{A}$, then

$$I_{\mathcal{L}_\mathcal{A}} = \{ \varphi : a \to b \mid \varphi \text{ factors through } \ell_b \}$$

is a two-sided ideal of morphisms in $\mathcal{A}$.

- We define the stable category of $\mathcal{A}$ by $\mathcal{L}_\mathcal{A}$ by

$$\text{Stab}_{\mathcal{L}_\mathcal{A}}(\mathcal{A}) := \mathcal{A}/I_{\mathcal{L}_\mathcal{A}},$$

i.e.,

1. The object class is same as that of $\mathcal{A}$.
2. For two objects $a, b \in \mathcal{A}$ it is

$$\text{Hom}_{\text{Stab}_{\mathcal{L}_\mathcal{A}}}(\mathcal{A})(a, b) := \text{Hom}_{\mathcal{A}}(a, b)/I_{\mathcal{L}_\mathcal{A}}(a, b),$$

where $I_{\mathcal{L}_\mathcal{A}}(a, b) := I_{\mathcal{L}_\mathcal{A}} \cap \text{Hom}_{\mathcal{A}}(a, b)$. 
A category $\mathcal{A}$ is called computable with a system of lifting objects $\mathcal{L}_\mathcal{A}$ if $\mathcal{A}$ is computable and is equipped with the following basic operations:

<table>
<thead>
<tr>
<th>Basic operation</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsLiftingObject</td>
<td>$b \in \mathcal{L}_\mathcal{A}$</td>
<td></td>
</tr>
<tr>
<td>LiftingObject</td>
<td>$b$</td>
<td>$L(b)$</td>
</tr>
<tr>
<td>MorphismFrom-LiftingObject</td>
<td>$b : \mathcal{L}_\mathcal{A}(b) \to b$</td>
<td></td>
</tr>
<tr>
<td>LiftingMorphism</td>
<td>$\phi : \mathcal{A}(a) \to \mathcal{A}(b)$</td>
<td></td>
</tr>
<tr>
<td>IsLiftableThroughLiftingObject</td>
<td>$\phi \in \mathcal{I}_\mathcal{A}(a,b)$</td>
<td></td>
</tr>
</tbody>
</table>
A category \( \mathcal{A} \) is called computable with a system of lifting objects \( \mathcal{L}_\mathcal{A} \) if \( \mathcal{A} \) is computable and is equipped with the following basic operations:

<table>
<thead>
<tr>
<th>Basic operation</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsLiftingObject</td>
<td>( b )</td>
<td>( b \in \mathcal{L}_\mathcal{A} ? )</td>
</tr>
</tbody>
</table>
A category $\mathcal{A}$ is called computable with a system of lifting objects $\mathcal{L}_\mathcal{A}$ if $\mathcal{A}$ is computable and is equipped with the following basic operations:

<table>
<thead>
<tr>
<th>Basic operation</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{IsLiftingObject}$</td>
<td>$b$</td>
<td>$b \in \mathcal{L}_\mathcal{A}$?</td>
</tr>
<tr>
<td>$\text{LiftingObject}$</td>
<td>$b$</td>
<td>$\mathcal{L}_b$</td>
</tr>
</tbody>
</table>
A category $\mathcal{A}$ is called computable with a system of lifting objects $\mathcal{L}_\mathcal{A}$ if $\mathcal{A}$ is computable and is equipped with the following basic operations:

<table>
<thead>
<tr>
<th>Basic operation</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsLiftingObject</td>
<td>$b$</td>
<td>$b \in \mathcal{L}_\mathcal{A}$?</td>
</tr>
<tr>
<td>LiftingObject</td>
<td>$b$</td>
<td>$\mathcal{L}_b$</td>
</tr>
<tr>
<td>MorphismFromLiftingObject</td>
<td>$b$</td>
<td>$\ell_b : \mathcal{L}_b \to b$</td>
</tr>
</tbody>
</table>
A category $\mathcal{A}$ is called computable with a system of lifting objects $\mathcal{L}_A$ if $\mathcal{A}$ is computable and is equipped with the following basic operations:

<table>
<thead>
<tr>
<th>Basic operation</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsLiftingObject</td>
<td>$b$</td>
<td>$b \in \mathcal{L}_A?$</td>
</tr>
<tr>
<td>LiftingObject</td>
<td>$b$</td>
<td>$\mathcal{L}_b$</td>
</tr>
<tr>
<td>MorphismFrom-LiftingObject</td>
<td>$b$</td>
<td>$\ell_b : \mathcal{L}_b \rightarrow b$</td>
</tr>
<tr>
<td>LiftingMorphism</td>
<td>$\varphi : a \rightarrow b$</td>
<td>$\mathcal{L}_\varphi : \mathcal{L}_a \rightarrow \mathcal{L}_b$</td>
</tr>
</tbody>
</table>
A category $\mathcal{A}$ is called computable with a system of lifting objects $\mathcal{L}_A$ if $\mathcal{A}$ is computable and is equipped with the following basic operations:

<table>
<thead>
<tr>
<th>Basic operation</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsLiftingObject $b$</td>
<td>$b$</td>
<td>$b \in \mathcal{L}_A$?</td>
</tr>
<tr>
<td>LiftingObject $b$</td>
<td>$b$</td>
<td>$\mathcal{L}_b$</td>
</tr>
<tr>
<td>MorphismFromLiftingObject $b$</td>
<td>$b$</td>
<td>$\ell_b : \mathcal{L}_b \rightarrow b$</td>
</tr>
<tr>
<td>LiftingMorphism $\varphi : a \rightarrow b$</td>
<td>$\mathcal{L}_\varphi : \mathcal{L}_a \rightarrow \mathcal{L}_b$</td>
<td></td>
</tr>
<tr>
<td>IsLiftatableThroughLiftingObject</td>
<td>$\varphi : a \rightarrow b$</td>
<td>$\varphi \in \mathcal{I}_{\mathcal{L}_A}(a, b)$?</td>
</tr>
</tbody>
</table>
Theorem

Let $\mathcal{A}$ be a computable additive category equipped with a lifting system $\mathcal{L}_A$.

If lifts are computable in $\mathcal{A}$, then $\text{Stab} \mathcal{L}_A(\mathcal{A})$ is computable additive.

The canonical projection functor $\mathcal{A} \to \text{Stab} \mathcal{L}_A(\mathcal{A})$ is additive.

If $\mathcal{A}$ has a $D$-homomorphism structure such that $D$ is abelian with a projective distinguished object, then $\text{Stab} \mathcal{L}_A(\mathcal{A})$ has a $D$-homomorphism structure.

Remark

All axioms and statements here can be dualized. The dual notation of a lifting object will be called a colifting object.
Theorem

Let $\mathcal{A}$ be a computable additive category equipped with a lifting system $\mathcal{L}_\mathcal{A}$.

If lifts are computable in $\mathcal{A}$, then $\text{Stab}_{\mathcal{L}_\mathcal{A}}(\mathcal{A})$ is computable additive.
Theorem

Let $\mathcal{A}$ be a computable additive category equipped with a lifting system $\mathcal{L}_\mathcal{A}$.

- If lifts are computable in $\mathcal{A}$, then $\text{Stab}_{\mathcal{L}_\mathcal{A}}(\mathcal{A})$ is computable additive.
- The canonical projection functor $\mathcal{A} \rightarrow \text{Stab}_{\mathcal{L}_\mathcal{A}}(\mathcal{A})$ is additive.
Computability of stable categories

Theorem

Let $\mathcal{A}$ be a computable additive category equipped with a lifting system $\mathcal{L}_\mathcal{A}$.

- If lifts are computable in $\mathcal{A}$, then $\text{Stab}_{\mathcal{L}_\mathcal{A}}(\mathcal{A})$ is computable additive.

- The canonical projection functor $\mathcal{A} \rightarrow \text{Stab}_{\mathcal{L}_\mathcal{A}}(\mathcal{A})$ is additive.

- If $\mathcal{A}$ has a $D$-homomorphism structure such that $D$ is abelian with a projective distinguished object, then $\text{Stab}_{\mathcal{L}_\mathcal{A}}(\mathcal{A})$ has a $D$-homomorphism structure.
Computability of stable categories

Theorem

Let \( \mathcal{A} \) be a computable additive category equipped with a lifting system \( \mathcal{L}_\mathcal{A} \).

- If lifts are computable in \( \mathcal{A} \), then \( \text{Stab}_{\mathcal{L}_\mathcal{A}}(\mathcal{A}) \) is computable additive.
- The canonical projection functor \( \mathcal{A} \to \text{Stab}_{\mathcal{L}_\mathcal{A}}(\mathcal{A}) \) is additive.
- If \( \mathcal{A} \) has a \( D \)-homomorphism structure such that \( D \) is abelian with a projective distinguished object, then \( \text{Stab}_{\mathcal{L}_\mathcal{A}}(\mathcal{A}) \) has a \( D \)-homomorphism structure.

Remark

All axioms and statements here can be dualized. The dual notation of a lifting object will be called a colifting object.
\textbf{CAP demo: finitely-generated abelian groups $\text{ab}$}

\begin{verbatim}
gap> ZZ := HomalgRingOfIntegers( );
\text{Z}

gap> ab := LeftPresentations( ZZ : FinalizeCategory := false );
\text{Category of left presentations of Z}

gap> AddMorphismFromLiftingObject( ab,
> EpimorphismFromSomeProjectiveObject );

gap> Finalize( ab );
true

gap> InfoOfInstalledOperationsOfCategory( ab );
44 primitive operations were used to derive 196 basic ones for
this ab Abelian category with enough projectives category

gap> CanCompute( ab, "LiftingMorphism" );
true

gap> IsAbelianCategoryWithEnoughProjectives( ab );
true
\end{verbatim}
\[ \mathbb{Z}, \]

\[
\text{gap> } \ZZ := \text{HomalgRingOfIntegers}( );
\]

\[ \mathbb{Z} \]
\[ \mathbb{Z}, \ \text{fpres}_\mathbb{Z} \]

\texttt{gap> ZZ := HomalgRingOfIntegers();}
\texttt{Z}

\texttt{gap> ab := LeftPresentations( ZZ : FinalizeCategory := false );}
\texttt{Category of left presentations of \textit{Z}}
\[
\begin{align*}
\mathbb{Z}, & \quad \text{fpres}_{\mathbb{Z}} \simeq \text{ab}, \\
\text{gap> } & \text{ZZ := HomalgRingOfIntegers( );} \\
\text{Z} & \\
\text{gap> ab := LeftPresentations( ZZ : FinalizeCategory := false );} \\
\text{Category of left presentations of } \mathbb{Z}
\end{align*}
\]
\[ \mathbb{Z}, \text{fpres}_\mathbb{Z} \cong \text{ab}, \]

\begin{verbatim}
gap> ZZ := HomalgRingOfIntegers( );
\mathbb{Z}

gap> ab := LeftPresentations( ZZ : FinalizeCategory := false );
Category of left presentations of \mathbb{Z}

gap> AddMorphismFromLiftingObject( ab, > EpimorphismFromSomeProjectiveObject );
\end{verbatim}
\[ \mathbb{Z}, \text{fpres}_\mathbb{Z} \cong \text{ab}, \]

\texttt{gap> ZZ := HomalgRingOfIntegers(); Z}

\texttt{gap> ab := LeftPresentations( ZZ : FinalizeCategory := false );}

\texttt{Category of left presentations of \texttt{Z}}

\texttt{gap> AddMorphismFromLiftingObject( ab,}

\texttt{> EpimorphismFromSomeProjectiveObject );}

\texttt{gap> Finalize( ab );}

\texttt{true}
\[ \mathbb{Z}, \text{fpres}_\mathbb{Z} \cong \text{ab}, \]

\begin{verbatim}
gap> ZZ := HomalgRingOfIntegers( );
\end{verbatim}
\begin{verbatim}
Z
\end{verbatim}

\begin{verbatim}
gap> ab := LeftPresentations( ZZ : FinalizeCategory := false );
\end{verbatim}
\begin{verbatim}
Category of left presentations of \( \mathbb{Z} \)
\end{verbatim}

\begin{verbatim}
gap> AddMorphismFromLiftingObject( ab, 
> EpimorphismFromSomeProjectiveObject );
\end{verbatim}
\begin{verbatim}
gap> Finalize( ab );
\end{verbatim}
\begin{verbatim}
true
\end{verbatim}

\begin{verbatim}
gap> InfoOfInstalledOperationsOfCategory( ab );
\end{verbatim}
\begin{verbatim}
44 primitive operations were used to derive 196 basic ones for this \text{ab} Abelian category with enough projectives category
\end{verbatim}
CAP demo: finitely-generated abelian groups \( \text{ab} \)

\[ \mathbb{Z}, \quad \text{fpres}_\mathbb{Z} \cong \text{ab}, \]

\[ \text{gap> } \text{ZZ} := \text{HomalgRingOfIntegers}(); \]
\[ \mathbb{Z} \]

\[ \text{gap> } \text{ab} := \text{LeftPresentations}( \text{ZZ} : \text{FinalizeCategory} := \text{false} ); \]
Category of left presentations of \( \mathbb{Z} \)

\[ \text{gap> } \text{AddMorphismFromLiftingObject}( \text{ab}, \>	ext{EpimorphismFromSomeProjectiveObject} ); \]

\[ \text{gap> } \text{Finalize}( \text{ab} ); \]
true

\[ \text{gap> } \text{InfoOfInstalledOperationsOfCategory}( \text{ab} ); \]
44 primitive operations were used to derive 196 basic ones for this \( \text{ab} \) Abelian category with enough projectives category

\[ \text{gap> } \text{CanCompute}( \text{ab}, "\text{LiftingMorphism}" ); \]
true
CAP demo: finitely-generated abelian groups \( ab \)

\[
\mathbb{Z}, \quad \text{fpres}_\mathbb{Z} \cong ab,
\]

\begin{verbatim}
gap> ZZ := HomalgRingOfIntegers( );
\mathbb{Z}

gap> ab := LeftPresentations( ZZ : FinalizeCategory := false );
Category of left presentations of \( \mathbb{Z} \)

gap> AddMorphismFromLiftingObject( ab,
> EpimorphismFromSomeProjectiveObject );

gap> Finalize( ab );
true

gap> InfoOfInstalledOperationsOfCategory( ab );
44 primitive operations were used to derive 196 basic ones for this \( ab \) Abelian category with enough projectives category

gap> CanCompute( ab, "LiftingMorphism" );
true

gap> IsAbelianCategoryWithEnoughProjectives( ab );
true
\end{verbatim}
CAP demo: stable category of \text{ab} by lifting objects

$$\mathbb{Z}, \text{fpres}_\mathbb{Z} \simeq \text{ab}, \text{Stab}_{\text{projs}}(\text{fpres}_\mathbb{Z})$$

\texttt{gap} > stable\_ab := \text{StableCategoryByLiftingStructure}(\text{ab});

The stable category of Category of left presentations of \mathbb{Z} ...
CAP demo: stable category of \textit{ab} by lifting objects

\[ \mathbb{Z}, \text{fpres}_\mathbb{Z} \simeq \text{ab}, \quad \text{Stab}_\text{projs}(\text{fpres}_\mathbb{Z}) \]

\texttt{gap> stable\_ab := StableCategoryByLiftingStructure( ab );}
\texttt{The stable category of Category of left presentations of \mathbb{Z} ...}

\texttt{gap> stable\_functor := CanonicalProjectionFunctor( stable\_ab );}
\texttt{Canonical projection functor ...}
\[ \mathbb{Z}, \text{fpres}_{\mathbb{Z}} \cong \text{ab}, \quad \text{Stab}_{\text{projs}}(\text{fpres}_{\mathbb{Z}}) \]

gap> stable_ab := StableCategoryByLiftingStructure( ab );
The stable category of Category of left presentations of \( \mathbb{Z} \) ...
gap> stable_functor := CanonicalProjectionFunctor( stable_ab );
Canonical projection functor ...
gap> m := HomalgMatrix( "[ [ 2, 3, 4 ], [ 3, 4, 3 ] ]", ZZ );;
\[ \mathbb{Z}, \text{fpres}_\mathbb{Z} \simeq \text{ab}, \quad \text{Stab}_{\text{projs}}(\text{fpres}_\mathbb{Z}) \]

```gap
gap> stable_ab := StableCategoryByLiftingStructure( ab );
The stable category of Category of left presentations of \( \mathbb{Z} \) ...
gap> stable_functor := CanonicalProjectionFunctor( stable_ab );
Canonical projection functor ...
gap> m := HomalgMatrix( "[ [ 2, 3, 4 ], [ 3, 4, 3 ] ]", ZZ );;
gap> a := AsLeftPresentation( m );
<An object in Category of left presentations of \( \mathbb{Z} \)>```

```
SmithNormalFormIntegerMat( [ [ 2, 3, 4 ], [ 3, 4, 3 ] ] );
[ [ 1, 0, 0 ], [ 0, 1, 0 ] ]
InfoOfInstalledOperationsOfCategory( stable_ab );
20 primitive operations were used to derive 73 basic ones ...
```
Z, \ text{fpres}_Z \simeq ab, \ \text{Stab}_{\text{projs}}(\text{fpres}_Z)

\text{gap> stable_ab := StableCategoryByLiftingStructure( ab );}
The stable category of Category of left presentations of Z ... 
\text{gap> stable_functor := CanonicalProjectionFunctor( stable_ab );}
Canonical projection functor ... 
\text{gap> m := HomalgMatrix( "[ [ 2, 3, 4 ], [ 3, 4, 3 ] ]", ZZ ); ;}
\text{gap> a := AsLeftPresentation( m );}
<An object in Category of left presentations of Z>
\text{gap> stable_a := ApplyFunctor( stable_functor, a );}
<An object in The stable category of Category of left presentations of Z ...>
\[ \mathbb{Z}, \text{fpres}_\mathbb{Z} \cong \text{ab}, \quad \text{Stab}_{\text{projs}}(\text{fpres}_\mathbb{Z}) \]

```gap
gap> stable_ab := StableCategoryByLiftingStructure( ab );
The stable category of Category of left presentations of \( \mathbb{Z} \) ...
gap> stable_functor := CanonicalProjectionFunctor( stable_ab );
Canonical projection functor ...
gap> m := HomalgMatrix( "[ [ 2, 3, 4 ], [ 3, 4, 3 ] ]", ZZ );;
gap> a := AsLeftPresentation( m );
<An object in Category of left presentations of \( \mathbb{Z} \)>
gap> stable_a := ApplyFunctor( stable_functor, a );
<An object in The stable category of Category of left presentations of \( \mathbb{Z} \) ...>
gap> List( [ a, stable_a ], IsZeroForObjects );
[ false, true ]
```
CAP demo: stable category of \( \text{ab} \) by lifting objects

\[ \mathbb{Z}, \quad \text{fpres}_\mathbb{Z} \cong \text{ab}, \quad \text{Stab}_{\text{projs}}(\text{fpres}_\mathbb{Z}) \]

\textbf{gap> } \text{stable}\_\text{ab} := \text{StableCategoryByLiftingStructure}( \text{ab} );
\text{The stable category of \text{Category of left presentations of } \mathbb{Z} \ldots}

\textbf{gap> } \text{stable}\_\text{functor} := \text{CanonicalProjectionFunctor}( \text{stable}\_\text{ab} );
\text{Canonical projection functor \ldots}

\textbf{gap> } m := \text{HomalgMatrix}( "[ [ 2, 3, 4 ], [ 3, 4, 3 ] ]", \text{ZZ} );;

\textbf{gap> } a := \text{AsLeftPresentation}( m );
<\text{An object in \text{Category of left presentations of } \mathbb{Z}>}

\textbf{gap> } \text{stable}\_a := \text{ApplyFunctor}( \text{stable}\_\text{functor}, \text{a} );
<\text{An object in \text{The stable category of \text{Category of left presentations of } \mathbb{Z}} \ldots>}

\textbf{gap> } \text{List}( [ \text{a, stable}\_a ], \text{IsZeroForObjects} );
[ \text{false, true} ]

\textbf{gap> } \text{SmithNormalFormIntegerMat}( [ [ 2, 3, 4 ], [ 3, 4, 3 ] ] );
[ [ 1, 0, 0 ], [ 0, 1, 0 ] ]
**CAP demo: stable category of \( \text{ab} \) by lifting objects**

\[
\mathbb{Z}, \quad \text{fpres}_\mathbb{Z} \cong \text{ab}, \quad \text{Stab}_{\text{projs}}(\text{fpres}_\mathbb{Z})
\]

```gap
gap> stable_ab := StableCategoryByLiftingStructure( \text{ab} );
The stable category of Category of left presentations of \( \mathbb{Z} \) ...
gap> stable_functor := CanonicalProjectionFunctor( stable_ab );
Canonical projection functor ...
gap> m := HomalgMatrix( "[ [ 2, 3, 4 ], [ 3, 4, 3 ] ]", \text{ZZ} );

gap> a := AsLeftPresentation( m );
<An object in Category of left presentations of \( \mathbb{Z} \)>

gap> stable_a := ApplyFunctor( stable_functor, a );
<An object in The stable category of Category of left presentations of \( \mathbb{Z} \) ...>

gap> List( [ a, stable_a ], IsZeroForObjects );
[ false, true ]

gap> SmithNormalFormIntegerMat( [ [ 2, 3, 4 ], [ 3, 4, 3 ] ] );
[ [ 1, 0, 0 ], [ 0, 1, 0 ] ]

gap> InfoOfInstalledOperationsOfCategory( stable_ab );
20 primitive operations were used to derive 73 basic ones ...
```
More examples

- We can compute the example of the previous slide for any commutative computable ring $R$. 
More examples

- We can compute the example of the previous slide for any commutative computable ring $R$.
- The same holds for the exterior algebra $E$ of a $\mathbb{Q}$-vector space.
We can compute the example of the previous slide for any commutative computable ring $R$.

The same holds for the exterior algebra $E$ of a $\mathbb{Q}$-vector space.

The same holds for the category of finite representations $\text{frep}_s(Q, I)$ of an acyclic quiver $Q$ with relations given by ideal $I$. 
Null-homotopic chain morphisms

Let $\mathcal{A}$ be an additive category and $A_\bullet$, $B_\bullet$ be objects in its category of complexes $\text{Ch}^b(\mathcal{A})$. Then the morphisms $(\phi_i := h_i d_B i + 1 + d_A i h_i - 1 : A_i \to B_i)_{i \in \mathbb{Z}}$ define a chain morphism $\phi_\bullet : A_\bullet \to B_\bullet$. Such chain morphisms are called null-homotopic.
Let $\mathcal{A}$ be an additive category and $A_\bullet, B_\bullet$ be objects in its category of complexes $\text{Ch}^b(\mathcal{A})$ and let $(h_i : A_i \rightarrow B_{i+1})_{i \in \mathbb{Z}}$ be a family of morphisms in $\mathcal{A}$:

$$
\begin{array}{cccccccc}
A_\bullet: & \cdots & \leftarrow & A_{i-1} & \leftarrow & A_i & \leftarrow & A_{i+1} & \leftarrow & \cdots \\
& & d^A_i & & d^A_{i+1} & & d^A_{i+2} & & \\
& h_{i-1} & & h_i & & & & & \\
B_\bullet: & \cdots & \leftarrow & B_{i-1} & \leftarrow & B_i & \leftarrow & B_{i+1} & \leftarrow & \cdots \\
& & d^B_i & & d^B_{i+1} & & d^B_{i+2} & & \\
\end{array}
$$

Then the morphisms $\left(\varphi_i := h_i d^B_{i+1} + d^A_i h_{i-1} : A_i \rightarrow B_i\right)_{i \in \mathbb{Z}}$ define a chain morphism $\varphi_\bullet: A_\bullet \rightarrow B_\bullet$. Such chain morphisms are called null-homotopic.
Null-homotopic chain morphisms

Let $\mathcal{A}$ be an additive category and $A_\bullet, B_\bullet$ be objects in its category of complexes $\text{Ch}^b(\mathcal{A})$ and let $(h_i : A_i \to B_{i+1})_{i \in \mathbb{Z}}$ be a family of morphisms in $\mathcal{A}$:

\[ A_\bullet : \cdots \leftarrow A_{i-1} \leftarrow A_i \leftarrow A_{i+1} \leftarrow \cdots \]

\[ B_\bullet : \cdots \leftarrow B_{i-1} \leftarrow B_i \leftarrow B_{i+1} \leftarrow \cdots \]

\[ \varphi_{i-1} \downarrow \quad \varphi_i \downarrow \quad \varphi_{i+1} \downarrow \]

\[ h_{i-1} \quad h_i \quad \]

Then the morphisms $(\varphi_i := h_id_{i+1}^B + d_i^Ah_{i-1} : A_i \to B_i)_{i \in \mathbb{Z}}$ define a chain morphism $\varphi_\bullet : A_\bullet \to B_\bullet$. 

Such chain morphisms are called null-homotopic.
Null-homotopic chain morphisms

Let $\mathcal{A}$ be an additive category and $A_\bullet$, $B_\bullet$ be objects in its category of complexes $\text{Ch}^b(\mathcal{A})$ and let $(h_i : A_i \to B_{i+1})_{i \in \mathbb{Z}}$ be a family of morphisms in $\mathcal{A}$:

\[
A_\bullet : \cdots \leftarrow A_{i-1} \leftarrow A_i \leftarrow A_{i+1} \leftarrow \cdots
\]
\[
B_\bullet : \cdots \leftarrow B_{i-1} \leftarrow B_i \leftarrow B_{i+1} \leftarrow \cdots
\]

Then the morphisms $(\varphi_i := h_id_{i+1}^B + d_i^Ah_{i-1} : A_i \to B_i)_{i \in \mathbb{Z}}$ define a chain morphism $\varphi_\bullet : A_\bullet \to B_\bullet$. Such chain morphisms are called null-homotopic.
Mapping cone

For a chain morphism \( \varphi_\bullet : A_\bullet \rightarrow B_\bullet \) we define the cone of \( \varphi_\bullet \), denoted by \( \text{Cone}(\varphi_\bullet) \), to be the following complex:

\[
\cdots \quad A_{i-2} \oplus B_{i-1} \quad \overset{d_i^{\text{Cone}(\varphi_\bullet)}}{\longrightarrow} \quad A_{i-1} \oplus B_i \quad \overset{}{\longleftarrow} \quad \cdots
\]
For a chain morphism \( \varphi \colon A \to B \) we define the cone of \( \varphi \), denoted by \( \text{Cone}(\varphi) \), to be the following complex:

\[
\cdots \leftarrow A_{i-2} \oplus B_{i-1} \xleftarrow{d^\text{Cone}(\varphi)_i} A_{i-1} \oplus B_i \xleftarrow{d^\text{Cone}(\varphi)_i} \cdots
\]

where \( d^\text{Cone}(\varphi)_i \) for \( i \in \mathbb{Z} \) is given by the following matrix:

\[
\begin{bmatrix}
d^A_{i-1} & -\varphi_{i-1} \\
0 & d^B_i
\end{bmatrix}.
\]
For a chain morphism $\varphi_\bullet : A_\bullet \to B_\bullet$ we define the cone of $\varphi_\bullet$, denoted by $\text{Cone}(\varphi_\bullet)$, to be the following complex:

$$\cdots \leftarrow A_{i-2} \oplus B_{i-1} \overset{d_i^{\text{Cone}(\varphi_\bullet)}}{\longrightarrow} A_{i-1} \oplus B_i \leftarrow \cdots$$

where $d_i^{\text{Cone}(\varphi_\bullet)}$ for $i \in \mathbb{Z}$ is given by the following matrix:

$$
\begin{bmatrix}
  d^A_{i-1} & -\varphi_{i-1} \\
  0 & d^B_i
\end{bmatrix}.
$$

Moreover, we have the obvious natural injection of $B_\bullet$ in $\text{Cone}(\varphi_\bullet)$, usually denoted by $\iota_{\varphi_\bullet}$. 
A chain morphism \( \varphi : A \to B \) is null-homotopic iff \( \iota_{\text{id}(A)} \) is coliftable along \( \varphi \), i.e., there is \( \delta \) such that the following diagram commutes:

\[
\begin{array}{ccc}
A & \xrightarrow{\varphi} & B \\
\downarrow{\iota_{\text{id}(A)}} & & \uparrow{\delta} \\
\text{Cone}(\text{id}_{A}) & & \\
\end{array}
\]

The following code in CAP checks whether \( \varphi \) is null-homotopic:

```bash
gap> A := Source( phi );
gap> id_A := IdentityMorphism( A );
gap> iota := NaturalInjectionInMappingCone( id_A );
gap> IsColiftable( iota, phi );
or
gap> IsNullHomotopic( phi );
```
A chain morphism $\varphi : A \to B$ is null-homotopic iff $\iota_{\text{id}(A)}$ is cofibrable along $\varphi$, i.e., there is $\delta$ such that the following diagram commutes:

\[
\begin{array}{ccc}
A & \xrightarrow{\varphi} & B \\
\downarrow{\iota_{\text{id}(A)}} & & \downarrow{\delta} \\
\text{Cone(}\text{id}_A) & & \\
\end{array}
\]

The following code in CAP checks whether $\varphi$ is null-homotopic:

```gap
gap> A := Source( phi ); id_A := IdentityMorphism( A );
gap> iota := NaturalInjectionInMappingCone( id_A );
gap> IsColiftable( iota, phi );
```

or

```gap
gap> IsNullHomotopic( phi );
```
A chain morphism $\varphi : A \to B$ is null-homotopic iff $\iota_{\text{id}(A)}$ is coliftable along $\varphi$, i.e., there is $\delta$ such that the following diagram commutes:

\[
\begin{array}{ccc}
A & \xrightarrow{\varphi} & B \\
\downarrow{\iota_{\text{id}(A)}} & & \downarrow{\delta} \\
\text{Cone}(\text{id}_A) & & \\
\end{array}
\]

The following code in CAP checks whether $\varphi$ is null-homotopic:

```gap
gap> A := Source( phi ); id_A := IdentityMorphism( A );
gap> iota := NaturalInjectionInMappingCone( id_A );
gap> IsColiftable( iota, phi );
or
gap> IsNullHomotopic( phi );
```
A chain morphism $\varphi_\bullet : A_\bullet \to B_\bullet$ is null-homotopic iff $\iota_{\text{id}(A_\bullet)}$ is coliftable along $\varphi_\bullet$, i.e., there is $\delta$ such that the following diagram commutes:

\[
\begin{array}{ccc}
A_\bullet & \xrightarrow{\varphi_\bullet} & B_\bullet \\
\downarrow{\iota_{\text{id}(A_\bullet)}} & & \leftarrow{\delta} \\
\text{Cone(id}_{A_\bullet}) & & \\
\end{array}
\]

The following code in CAP checks whether $\varphi$ is null-homotopic:

```gap
gap> A := Source( phi ); id_A := IdentityMorphism( A );
gap> iota := NaturalInjectionInMappingCone( id_A );
gap> IsColiftable( iota, phi );
```

or

```gap
gap> IsNullHomotopic( phi );
```
A chain morphism $\varphi : A \rightarrow B$ is null-homotopic iff $\iota_{\text{id}(A)}$ is coliftable along $\varphi$, i.e., there is $\delta$ such that the following diagram commutes:

\[
\begin{array}{ccc}
A & \xrightarrow{\varphi} & B \\
\downarrow{\iota_{\text{id}(A)}} & \circ & \\
\text{Cone}(\text{id}_{A}) & \xrightarrow{} & \\
\end{array}
\]

The following code in CAP checks whether $\varphi$ is null-homotopic:

\[
\begin{align*}
gap> & \ A := \text{Source}(\ phi \ ); \ \text{id}_A := \text{IdentityMorphism}(\ A \ ); \\
gap> & \ \text{iota} := \text{NaturalInjectionInMappingCone}(\ \text{id}_A \ ); \\
gap> & \ \text{IsColiftable}(\ \text{iota}, \ \phi \ ); \\
\end{align*}
\]

or

\[
\begin{align*}
gap> & \ \text{IsNullHomotopic}(\ \phi \ ); \\
\end{align*}
\]
Lemma

The class $\mathcal{C}$ of split exact complexes is a system of colifting objects for $\text{Ch}^b(A)$, explicitly $\mathcal{C}_{A^\bullet} := \text{Cone}(\text{id}_{A^\bullet})$ and $c_{A^\bullet} := \iota_{\text{id}_{A^\bullet}} : A^\bullet \to \mathcal{C}_{A^\bullet}$.
The bounded homotopy category

Lemma

The class $\mathcal{C}$ of split exact complexes is a system of colifting objects for $\text{Ch}^b(\mathcal{A})$, explicitly $\mathcal{C}_{A_\bullet} := \text{Cone}(\text{id}_{A_\bullet})$ and $c_{A_\bullet} := \iota_{\text{id}_{A_\bullet}} : A_\bullet \rightarrow \mathcal{C}_{A_\bullet}$.

Definition

Let $\mathcal{A}$ be an additive category, then the bounded homotopy category of $\mathcal{A}$, denoted by $\text{Ho}^b(\mathcal{A})$ is defined by

$$\text{Ho}^b(\mathcal{A}) := \text{Stab}_C(\text{Ch}^b(\mathcal{A})).$$
Lemma

The class $\mathcal{C}$ of split exact complexes is a system of colifting objects for $\text{Ch}^b(\mathcal{A})$, explicitly $\mathcal{C}_{A_\bullet} := \text{Cone}(\text{id}_{A_\bullet})$ and $c_{A_\bullet} := \iota_{\text{id}_{A_\bullet}} : A_\bullet \to \mathcal{C}_{A_\bullet}$.

Definition

Let $\mathcal{A}$ be an additive category, then the bounded homotopy category of $\mathcal{A}$, denoted by $\text{Ho}^b(\mathcal{A})$ is defined by

$$\text{Ho}^b(\mathcal{A}) := \text{Stab}_C(\text{Ch}^b(\mathcal{A})).$$

Theorem

Let $\mathcal{A}$ be a computable additive category with $D$-homomorphism structure such that $D$ is abelian with a projective distinguished object. Then $\text{Ho}^b(\mathcal{A})$ is computable additive and has a $D$-homomorphism structure.
Let us go back to our first example

```gap
gap> ZZ := HomalgRingOfIntegers( );
Z
```
Let us go back to our first example

\begin{verbatim}
gap> ZZ := HomalgRingOfIntegers( );
Z

gap> ab := LeftPresentations( ZZ );
Category of left presentations of Z
\end{verbatim}
Let us go back to our first example

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

gap> ab := LeftPresentations( ZZ );
Category of left presentations of Z

gap> H := HomotopyCategory( ab );
Homotopy category of Category of left presentations of Z
```
Let us go back to our first example

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

gap> ab := LeftPresentations( ZZ );
Category of left presentations of Z

gap> H := HomotopyCategory( ab );
Homotopy category of Category of left presentations of Z

gap> InfoOfInstalledOperationsOfCategory( H );
32 primitive operations were used to derive 92 basic ones for this ab additive category
```
Let us go back to our first example

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

gap> ab := LeftPresentations( ZZ );
Category of left presentations of Z

gap> H := HomotopyCategory( ab );
Homotopy category of Category of left presentations of Z

gap> InfoOfInstalledOperationsOfCategory( H );
32 primitive operations were used to derive 92 basic ones for this ab additive category

gap> F := CanonicalProjectionFunctor( H );
Canonical projection functor from Chain complexes category over Category of left presentations of Z in Homotopy category of Category of left presentations of Z
```
CAP demo: the bounded homotopy category of ab

\[
\begin{align*}
C & : 0 \leftarrow \mathbb{Z}^{1 \times 2} \leftarrow \begin{pmatrix} 1 & 3 \\ \end{pmatrix} \mathbb{Z}^{1 \times 1} \leftarrow 0 \\
& \downarrow \begin{pmatrix} 26 & 13 \\ 13 & 13 \\ \end{pmatrix} \\
D & : 0 \leftarrow \mathbb{Z}^{1 \times 2} \leftarrow \begin{pmatrix} 2 & 3 \\ 5 & 1 \\ \end{pmatrix} \mathbb{Z}^{1 \times 2} \leftarrow 0
\end{align*}
\]
CAP demo: the bounded homotopy category of \( \text{ab} \)

\[
\begin{align*}
C : \quad 0 & \overset{}{\longleftarrow} \mathbb{Z}^{1 \times 2} & \overset{(1, 3)}{\longleftarrow} \mathbb{Z}^{1 \times 1} & \overset{}{\longleftarrow} 0 \\
& \quad \quad \quad \quad \quad \downarrow \begin{pmatrix} 26 & 13 \\ 13 & 13 \end{pmatrix} \quad \downarrow \begin{pmatrix} 2 & 3 \\ 5 & 1 \end{pmatrix} \quad \downarrow \begin{pmatrix} 15 & 7 \end{pmatrix} \\
D : \quad 0 & \overset{}{\longleftarrow} \frac{\mathbb{Z}^{1 \times 2}}{\langle (0, 2) \rangle} & \overset{}{\longleftarrow} \mathbb{Z}^{1 \times 2} & \overset{}{\longleftarrow} 0
\end{align*}
\]

\[
gap> C_1 := \text{FreeLeftPresentation}(1, \mathbb{Z});;
\]
\[
gap> C_0 := \text{DirectSum}(C_1, C_1);
\]
\[
gap> m := \text{HomalgMatrix}([[1, 3]]);;
\]
\[
gap> dC_1 := \text{PresentationMorphism}(C_1, m, C_0);
\]
\[
gap> C := \text{ChainComplex}([dC_1], 1);
\]
\[
gap> D_1 := C_0;
\]
\[
gap> D_0 := \text{AsLeftPresentation}(\text{HomalgMatrix}([[0, 2]]));
\]
\[
gap> m := \text{HomalgMatrix}([[2, 3], [5, 1]]);
\]
\[
dD_1 := \text{PresentationMorphism}(D_1, m, D_0);
\]
\[
D := \text{ChainComplex}([dD_1], 1);
\]
CAP demo: the bounded homotopy category of ab

\[
\begin{array}{cccccc}
C: & 0 & \rightarrow & \mathbb{Z}^{1 \times 2} & \rightarrow & \mathbb{Z}^{1 \times 1} & \rightarrow & 0 \\
& & \downarrow & \left( \begin{array}{cc} 1 & 3 \\ 26 & 13 \\ 13 & 13 \end{array} \right) & \downarrow & \left( \begin{array}{cc} 2 & 3 \\ 5 & 1 \end{array} \right) & \downarrow & \left( \begin{array}{cc} 15 & 7 \end{array} \right) & \rightarrow & 0 \\
D: & 0 & \rightarrow & \mathbb{Z}^{1 \times 2} & \rightarrow & \mathbb{Z}^{1 \times 2} & \rightarrow & 0 \\
& & \langle (0, 2) \rangle & \rightarrow & \langle (0, 2) \rangle & \rightarrow & 0 \\
\end{array}
\]

gap> C_1 := FreeLeftPresentation( 1, ZZ );;
gap> C_0 := DirectSum( C_1, C_1 );;
gap> m := HomalgMatrix( \[ [1,3] \], ZZ );;
gap> dC_1 := PresentationMorphism( C_1, m, C_0 );;
gap> C := ChainComplex( [ dC_1 ], 1 );
\[ C : \begin{array}{c} 0 \end{array} \xleftarrow{\begin{pmatrix} 26 & 13 \\ 13 & 13 \end{pmatrix}} \begin{array}{c} \mathbb{Z}^{1 \times 2} \end{array} \xleftarrow{\begin{pmatrix} 1 & 3 \end{pmatrix}} \begin{array}{c} \mathbb{Z}^{1 \times 1} \end{array} \xleftarrow{\begin{pmatrix} 2 & 3 \\ 5 & 1 \end{pmatrix}} \begin{array}{c} \mathbb{Z}^{1 \times 2} \end{array} \xleftarrow{\begin{pmatrix} 15 & 7 \end{pmatrix}} \begin{array}{c} 0 \end{array} \]

\[ D : \begin{array}{c} 0 \end{array} \xleftarrow{\begin{array}{c} \langle (0, 2) \rangle \end{array}} \begin{array}{c} \mathbb{Z}^{1 \times 2} \end{array} \xleftarrow{\begin{pmatrix} 15 & 7 \end{pmatrix}} \begin{array}{c} \mathbb{Z}^{1 \times 1} \end{array} \xleftarrow{\begin{array}{c} \langle (0, 2) \rangle \end{array}} \begin{array}{c} \mathbb{Z}^{1 \times 2} \end{array} \xleftarrow{\begin{pmatrix} 15 & 7 \end{pmatrix}} \begin{array}{c} 0 \end{array} \]

\text{gap> } C_1 := \text{FreeLeftPresentation}(1, \mathbb{Z}Z);;
\text{gap> } C_0 := \text{DirectSum}(C_1, C_1) ;;
\text{gap> } m := \text{HomalgMatrix}(\begin{bmatrix}1 & 3\end{bmatrix}, \mathbb{Z}Z) ;;
\text{gap> } dC_1 := \text{PresentationMorphism}(C_1, m, C_0) ;;
\text{gap> } C := \text{ChainComplex}(\begin{bmatrix}dC_1\end{bmatrix}, 1) ;;
\text{gap> } D_1 := C_0;
\text{gap> } D_0 := \text{AsLeftPresentation}(\text{HomalgMatrix}(\begin{bmatrix}0 & 2\end{bmatrix}, \mathbb{Z}Z) );
CAP demo: the bounded homotopy category of $\text{ab}$

\[
\begin{align*}
\text{C: } & 0 \leftarrow \mathbb{Z}^{1 \times 2} \leftarrow \begin{pmatrix} 1 & 3 \\ 26 & 13 \\ 13 & 13 \end{pmatrix} \mathbb{Z}^{1 \times 1} \leftarrow 0 \\
\text{D: } & 0 \leftarrow \mathbb{Z}^{1 \times 2} \left\langle \begin{pmatrix} 0 & 2 \end{pmatrix} \right\rangle \leftarrow \begin{pmatrix} 2 & 3 \\ 5 & 1 \end{pmatrix} \mathbb{Z}^{1 \times 2} \leftarrow 0
\end{align*}
\]

\[
gap> \text{C}_1 := \text{FreeLeftPresentation}(1, \mathbb{Z});;
\]
\[
gap> \text{C}_0 := \text{DirectSum}(\text{C}_1, \text{C}_1);
\]
\[
gap> \text{m} := \text{HomalgMatrix}(\begin{bmatrix} 1 & 3 \\ 0 & 2 \end{bmatrix}, \mathbb{Z});;
\]
\[
gap> \text{dC}_1 := \text{PresentationMorphism}(\text{C}_1, \text{m}, \text{C}_0);
\]
\[
gap> \text{C} := \text{ChainComplex}(\begin{bmatrix} \text{dC}_1 \end{bmatrix}, 1);
\]
\[
gap> \text{D}_1 := \text{C}_0;
\]
\[
gap> \text{D}_0 := \text{AsLeftPresentation}(\text{HomalgMatrix}(\begin{bmatrix} 0 & 2 \end{bmatrix}, \mathbb{Z}));;
\]
\[
gap> \text{m} := \text{HomalgMatrix}(\begin{bmatrix} 2 & 3 \\ 5 & 1 \end{bmatrix}, \mathbb{Z});;
\]
\[
gap> \text{dD}_1 := \text{PresentationMorphism}(\text{D}_1, \text{m}, \text{D}_0);
\]
\[
gap> \text{D} := \text{ChainComplex}(\begin{bmatrix} \text{dD}_1 \end{bmatrix}, 1);
\]
\[ C : 0 \leftarrow \mathbb{Z}^{1\times2} \leftarrow \begin{pmatrix} 1 & 3 \\ \end{pmatrix} \mathbb{Z}^{1\times1} \leftarrow 0 \]
\[ D : 0 \leftarrow \mathbb{Z}^{1\times2} \leftarrow \begin{pmatrix} 2 & 3 \\ 5 & 1 \\ \end{pmatrix} \mathbb{Z}^{1\times2} \leftarrow 0 \]

\begin{align*}
gap> m := \text{HomalgMatrix}( [[26,13],[13,13]], \mathbb{Z} );;
gap> \phi_0 := \text{PresentationMorphism}( C_0, m, D_0 );
\end{align*}
\[
\begin{align*}
C: & \quad 0 \leftarrow \mathbb{Z}^{1 \times 2} \leftarrow \mathbb{Z}^{1 \times 1} \leftarrow 0 \\
& \quad \begin{pmatrix} 1 & 3 \\ 1 & 2 \end{pmatrix} \\
D: & \quad 0 \leftarrow \mathbb{Z}^{1 \times 2} \leftarrow \mathbb{Z}^{1 \times 2} \leftarrow 0 \\
& \quad \begin{pmatrix} 2 & 3 \\ 5 & 1 \end{pmatrix} \\
& \quad \begin{pmatrix} 15 & 7 \end{pmatrix}
\end{align*}
\]

gap> m := HomalgMatrix( [[26,13],[13,13]], ZZ );;
gap> phi_0 := PresentationMorphism( C_0, m, D_0 );
gap> m := HomalgMatrix( [[15,7]], ZZ );;
gap> phi_1 := PresentationMorphism( C_1, m, D_1 );
\textbf{CAP demo: the bounded homotopy category of ab}

\begin{equation*}
\begin{array}{cccccc}
C: & 0 & \xleftarrow{\mathbb{Z}^{1 \times 2}} & \mathbb{Z}^{1 \times 1} & \xleftarrow{\begin{pmatrix} 1 & 3 \\ 26 & 13 \\ 13 & 13 \end{pmatrix}} & D: & 0 \\
\xleftarrow{\begin{pmatrix} 26 & 13 \\ 13 & 13 \end{pmatrix}} & \mathbb{Z}^{1 \times 2} & \xleftarrow{\begin{pmatrix} 2 & 3 \\ 15 & 7 \end{pmatrix}} & \mathbb{Z}^{1 \times 2} & \xleftarrow{\{(0,2)\}} & \mathbb{Z}^{1 \times 1} & 0
\end{array}
\end{equation*}

\begin{verbatim}
gap> m := HomalgMatrix( [[26,13],[13,13]], ZZ );;
gap> phi_0 := PresentationMorphism( C_0, m, D_0 );;
gap> m := HomalgMatrix( [[15,7]], ZZ );;
gap> phi_1 := PresentationMorphism( C_1, m, D_1 );;
gap> phi := ChainMorphism( C, D, [ phi_0, phi_1 ], 0 );
gap> F_phi := ApplyFunctor( F, phi );
gap> IsZero( F_phi );
true
\end{verbatim}
CAP demo: the bounded homotopy category of \( \text{ab} \)

\[
\begin{align*}
C & : 0 \leftarrow \mathbb{Z}^{1 \times 2} \leftarrow \mathbb{Z}^{1 \times 1} \leftarrow 0 \\
\begin{pmatrix} 26 & 13 \\ 13 & 13 \end{pmatrix} & \downarrow \\
D & : 0 \leftarrow \mathbb{Z}^{1 \times 2} \leftarrow \mathbb{Z}^{1 \times 2} \leftarrow 0 \\
\begin{pmatrix} 2 & 3 \\ 5 & 1 \end{pmatrix} & \downarrow \\
\begin{pmatrix} 15 & 7 \end{pmatrix} & \downarrow \\
\end{align*}
\]

\[
\text{gap> } m := \text{HomalgMatrix( [[26,13],[13,13]], ZZ )};;
\text{gap> } \phi_0 := \text{PresentationMorphism( C_0, m, D_0 )};
\text{gap> } m := \text{HomalgMatrix( [[15,7]], ZZ )};;
\text{gap> } \phi_1 := \text{PresentationMorphism( C_1, m, D_1 )};
\text{gap> } \phi := \text{ChainMorphism( C, D, [ phi_0, phi_1 ], 0 )};
\text{gap> } F\_phi := \text{ApplyFunctor( F, phi )};
\text{gap> } \text{IsZero( F\_phi )};
\text{true}
\text{gap> } h\_morphisms := \text{HomotopyMorphisms( F\_phi )};
\text{<An infinite list>}
\]
\( \text{CAP demo: the bounded homotopy category of ab} \)

\[
\begin{array}{c}
\text{C: } 0 & \xleftarrow{\mathbb{Z}^1 \times 2} & \frac{(1 \ 3)}{26 \ 13} & \xleftarrow{\mathbb{Z}^1} & \frac{(2 \ 3)}{15 \ 7} & \xleftarrow{\mathbb{Z}^1 \times 2} & 0 \\
\text{D: } 0 & \xleftarrow{\mathbb{Z}^1 \times 2} & \frac{(13 \ 13)}{0 \ 2} & \xleftarrow{\mathbb{Z}^1} & \frac{(13 \ 13)}{0 \ 2} & \xleftarrow{\mathbb{Z}^1 \times 2} & 0
\end{array}
\]
CAP demo: the bounded homotopy category of \( \text{ab} \)

\[
\begin{align*}
C : 0 & \leftarrow \mathbb{Z}^{1 \times 2} \leftarrow \mathbb{Z}^{1 \times 1} & \left\langle \begin{pmatrix} 1 & 3 \\ 2 & 3 \end{pmatrix} \right\rangle \\
D : 0 & \leftarrow \mathbb{Z}^{1 \times 2} / \langle (0, 2) \rangle \left\langle \begin{pmatrix} 2 & 3 \\ 5 & 1 \end{pmatrix} \right\rangle & \left\langle \begin{pmatrix} 15 & 7 \end{pmatrix} \right\rangle
\end{align*}
\]

\texttt{gap> h0 := h\_morphisms[0];}
\texttt{<A morphism in Category of left presentations of \( \mathbb{Z} \)>}

\texttt{gap> IsCongruentForMorphisms( PreCompose( C^1, h0 ), phi[1] );}
\texttt{true}

\texttt{gap> IsCongruentForMorphisms( PreCompose( h0, D^1 ), phi[0] );}
\texttt{true}
CAP demo: the bounded homotopy category of ab

\[
\begin{array}{cccccc}
C: & 0 & \leftarrow & \mathbb{Z}^{1\times2} & \leftarrow & (1, 3) & \leftarrow & \mathbb{Z}^{1\times1} & \leftarrow & 0 \\
 & \leftarrow & \begin{pmatrix} 26 & 13 \\ 13 & 13 \end{pmatrix} & \downarrow & \begin{pmatrix} 2 & 3 \\ 5 & 1 \end{pmatrix} & \downarrow & \begin{pmatrix} 15 & 7 \end{pmatrix} & & \\
D: & 0 & \leftarrow & \mathbb{Z}^{1\times2}_{\langle (0, 2) \rangle} & \leftarrow & (1, 3, 13) & \leftarrow & (2, 3, 5, 1) & \leftarrow & 0 \\
\end{array}
\]

\text{gap} > \text{h0 := h_morphisms[0];}
\text{<A morphism in Category of left presentations of Z>}

\text{gap} > \text{Display( h0 );}
[ [ 633, -248 ],
 [ -206, 85 ] ]

\text{A morphism in Category of left presentations of Z}
\textbf{CAP demo: the bounded homotopy category of \textit{ab}}

\[
\begin{array}{c}
C: 0 \leftarrow \mathbb{Z}^{1 \times 2} \mathbb{Z}^{1\times 1} \leftarrow 0 \\
\begin{pmatrix}
26 & 13 \\
13 & 13
\end{pmatrix}
\downarrow
\begin{pmatrix}
2 & 3 \\
5 & 1
\end{pmatrix}
\downarrow
\begin{pmatrix}
15 & 7
\end{pmatrix}
\end{array}
\]

\[
D: 0 \leftarrow \mathbb{Z}^{1 \times 2} \mathbb{Z}^{1\times 2} \leftarrow 0 \\
\frac{\mathbb{Z}^{1 \times 2}}{\langle 0, 2 \rangle}
\]

\[\text{gap> h0 := h\_morphisms[ 0 ];}\]
\[\text{<A morphism in Category of left presentations of \textit{Z}>}\]

\[\text{gap> Display( h0 );}\]
\[\begin{array}{c}
[ [ 633, -248 ], \\
[ -206, 85 ]]
\end{array}\]

\[\text{A morphism in Category of left presentations of \textit{Z}}\]

\[\text{gap> IsCongruentForMorphisms( PreCompose( C^1, h0 ), phi[1] );}\]
\[\text{true}\]
CAP demo: the bounded homotopy category of \( \text{ab} \)

\[
\begin{array}{cccccc}
C : & 0 & \leftarrow & \mathbb{Z}^{1 \times 2} & \leftarrow & \mathbb{Z}^{1 \times 1} & \leftarrow & 0 \\
& \left( \begin{array}{cc}
26 & 13 \\
13 & 13 \\
\end{array} \right) & \downarrow & \left( \begin{array}{cc}
1 & 3 \\
2 & 3 \\
\end{array} \right) & \downarrow & \left( \begin{array}{cc}
15 & 7 \\
5 & 1 \\
\end{array} \right) & \\
D : & 0 & \leftarrow & \mathbb{Z}^{1 \times 2} / \langle \langle 0, 2 \rangle \rangle & \leftarrow & \mathbb{Z}^{1 \times 2} & \leftarrow & 0 \\
\end{array}
\]

\[
\text{gap> } h0 := h\_morphisms[0]; \\
\langle \text{A morphism in Category of left presentations of } \mathbb{Z} \rangle
\]

\[
\text{gap> } \text{Display}(h0); \\
[ [ 633, -248 ], \\
[ -206, 85 ] ]
\]

A morphism in Category of left presentations of \( \mathbb{Z} \)

\[
\text{gap> } \text{IsCongruentForMorphisms( PreCompose( C\^1, h0 ), phi[1] );} \\
\text{true}
\]

\[
\text{gap> } \text{IsCongruentForMorphisms( PreCompose( h0, D\^1 ), phi[0] );} \\
\text{true}
\]
For an acyclic quiver $Q$ with relations given by ideal $I$, the category $\text{Ho}^b(\text{freps}_Q(Q, I))$ is computable additive with $\text{Rows}_Q$-homomorphism structure.
For an acyclic quiver $Q$ with relations given by ideal $I$, the category $\text{Ho}^b(\text{freps}_Q(Q, I))$ is computable additive with $\text{Rows}_Q$-homomorphism structure.

For any commutative computable ring $S$, the category $\text{Ho}^b(\text{fpres}_S)$ is computable additive with $\text{fpres}_S$-homomorphism structure.
For an acyclic quiver $Q$ with relations given by ideal $I$, the category $\text{Ho}^b(\text{freps}_Q(Q, I))$ is computable additive with $\text{Rows}_Q$-homomorphism structure.

For any commutative computable ring $S$, the category $\text{Ho}^b(\text{fpres}_S)$ is computable additive with $\text{fpres}_S$-homomorphism structure.

For the $\mathbb{Z}_{\geq 0}$-graded, $\mathbb{Z}_{\leq 0}$-graded exterior algebra $E$ of $Q^n$, the categories $\text{Stab}_{\text{projs}}(E\text{-gfpres})$ and $\text{Ho}^b(E\text{-gfpres})$ are computable additive with $\text{Rows}_Q$-homomorphism structure.
Some bounded derived categories are equivalent to bounded homotopy categories, for instance $D^b(\text{fpre} \text{s}_R)$ for any computable ring $R$ with finite global dimension.
Applications

- Some bounded derived categories are equivalent to bounded homotopy categories, for instance $\mathcal{D}^b(f\text{pres}_R)$ for any computable ring $R$ with finite global dimension.
- Equivalences of categories can help answering many questions.
Applications

Some bounded derived categories are equivalent to bounded homotopy categories, for instance $\mathcal{D}^b(\text{fpres}_R)$ for any computable ring $R$ with finite global dimension.

Equivalences of categories can help answering many questions. A very famous example for this is

$$\mathcal{D}^b(n\text{-Beilinson quiver}) \cong \mathcal{D}^b(\mathbb{P}^n) \cong \text{Stab}_{\text{projs}}(E\text{-gfpres}).$$
Try out CAP interactively:
https://sebastianpos.github.io/Try-out-CAP/

CAP on GitHub:
https://github.com/homalg-project/CAP_project

