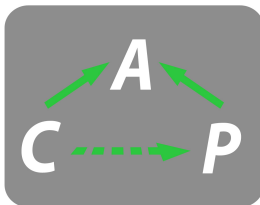


# A constructive approach to Freyd categories

Sebastian Posur

University of Siegen

April 26, 2018



# Finitely presented functors

# Finitely presented functors

Let  $R$  be a ring.

# Finitely presented functors

Let  $R$  be a ring.

**Q:** Is it possible to model the category of finitely presented functors

$$R\text{-Mod} \longrightarrow \mathbf{Ab}$$

on the computer?

# Finitely presented functors

Let  $R$  be a ring.

**Q:** Is it possible to model the category of finitely presented functors

$$R\text{-Mod} \longrightarrow \mathbf{Ab}$$

on the computer?

What are finitely presented functors?

# Finitely presented functors

Let  $R$  be a ring.

# Finitely presented functors

Let  $R$  be a ring.

## Definition

A functor  $F : R\text{-Mod} \rightarrow \mathbf{Ab}$  is called **finitely presented** if there exist  $A, B \in R\text{-Mod}$  and an exact sequence of functors

$$0 \longleftarrow F \longleftarrow \text{Hom}(B, -) \longleftarrow \text{Hom}(A, -)$$

# Finitely presented functors

Let  $R$  be a ring.

Q: Is it possible to model the category of finitely presented functors

$$R\text{-Mod} \longrightarrow \mathbf{Ab}$$

on the computer?



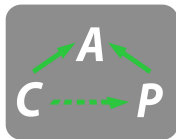
# Finitely presented functors

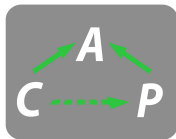
Let  $R$  be a ring.

**Q:** Is it possible to model the category of finitely presented functors

$$R\text{-fpmod} \longrightarrow \mathbf{Ab}$$

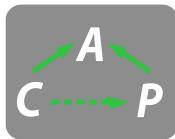
on the computer?





CAP - Categories, Algorithms, Programming  
(Gutsche, P., Skartsæterhagen)

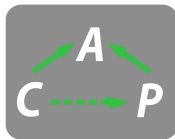
CAP is a software project in GAP facilitating the implementation of



## CAP - Categories, Algorithms, Programming (Gutsche, P., Skartsæterhagen)

CAP is a software project in GAP facilitating the implementation of

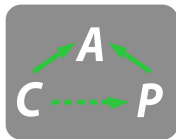
- 1 specific instances of categories,



## CAP - Categories, Algorithms, Programming (Gutsche, P., Skartsæterhagen)

CAP is a software project in GAP facilitating the implementation of

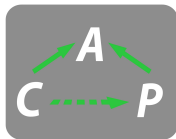
- 1 specific instances of categories,
- 2 category constructors,



## CAP - Categories, Algorithms, Programming (Gutsche, P., Skartsæterhagen)

CAP is a software project in GAP facilitating the implementation of

- 1 specific instances of categories,
- 2 category constructors,
- 3 categorical algorithms.



## CAP - Categories, Algorithms, Programming (Gutsche, P., Skartsæterhagen)

CAP is a software project in GAP facilitating the implementation of

- 1 specific instances of categories,
- 2 category constructors,
- 3 categorical algorithms.

# Category of finite dim. vector spaces

Let  $k$  be a field.



# Category of finite dim. vector spaces

Let  $k$  be a field.

## Category of finite dim. vector spaces

- $\text{Obj} := \text{finite dim. } k\text{-vector spaces}$

# Category of finite dim. vector spaces

Let  $k$  be a field.

## Category of finite dim. vector spaces

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

# Category of finite dim. vector spaces

Let  $k$  be a field.

## Category of finite dim. vector spaces

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

## Category of matrices

- $\text{Obj} := \mathbb{N}_0$

# Category of finite dim. vector spaces

Let  $k$  be a field.

## Category of finite dim. vector spaces

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

## Category of matrices

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

# Category of finite dim. vector spaces

Let  $k$  be a field.

## Category of finite dim. vector spaces

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

## Category of matrices

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

# Category of finite dim. vector spaces

Let  $k$  be a field.

## Category of finite dim. vector spaces

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

$\simeq$

## Category of matrices

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

# Category of finite dim. vector spaces

Let  $k$  be a field.

## Category of finite dim. vector spaces

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

$\simeq$

## Category of matrices (computerfriendly model)

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

# Category of finite dim. vector spaces

Let  $k$  be a field.

## Category of finite dim. vector spaces

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

$\simeq$

## Category of matrices (computerfriendly model)

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



# The language of category theory

Some categorical operations in abelian categories

# The language of category theory

## Some categorical operations in abelian categories

- $\oplus : \text{Obj} \times \text{Obj} \rightarrow \text{Obj}$

# The language of category theory

## Some categorical operations in abelian categories

- $\oplus : \text{Obj} \times \text{Obj} \rightarrow \text{Obj}$
- $+, - : \text{Hom}(A, B) \times \text{Hom}(A, B) \rightarrow \text{Hom}(A, B)$

# The language of category theory

## Some categorical operations in abelian categories

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

# The language of category theory

## Some categorical operations in abelian categories

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

# Implementation of the kernel

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

# Implementation of the kernel

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

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

# Implementation of the kernel

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

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



# Implementation of the kernel

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

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

$\ker \varphi$

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

# Implementation of the kernel

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

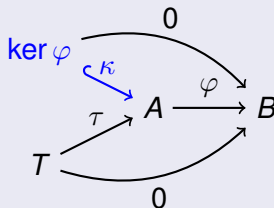
$\dots$  one needs an object  $\text{ker } \varphi$ ,  
its embedding  $\kappa = \text{KernelEmbedding}(\varphi)$ ,

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

# Implementation of the kernel

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

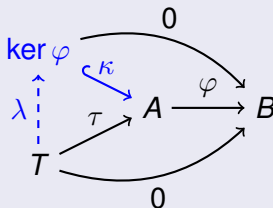
$\dots$  one needs an object  $\text{ker } \varphi$ ,  
its embedding  $\kappa = \text{KernelEmbedding}(\varphi)$ ,  
and for every test morphism  $\tau$



# Implementation of the kernel

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

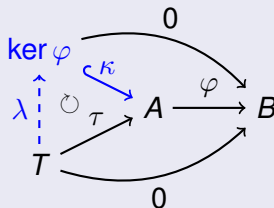
$\dots$  one needs an object  $\text{ker } \varphi$ ,  
its embedding  $\kappa = \text{KernelEmbedding}(\varphi)$ ,  
and for every test morphism  $\tau$   
a *unique* morphism  $\lambda = \text{KernelLift}(\varphi, \tau)$



# Implementation of the kernel

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

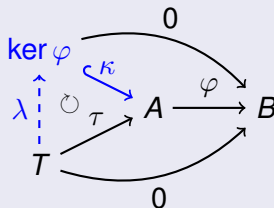
$\dots$  one needs an object  $\text{ker } \varphi$ ,  
its embedding  $\kappa = \text{KernelEmbedding}(\varphi)$ ,  
and for every test morphism  $\tau$   
a *unique* morphism  $\lambda = \text{KernelLift}(\varphi, \tau)$ , such that



# Implementation of the kernel

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

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

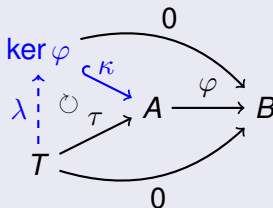


**3 algorithms needed for the kernel**

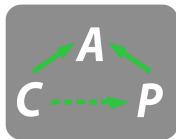
# Implementation of the kernel

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

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



**3 algorithms needed for the kernel/cokernel.**

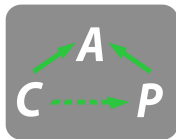


## CAP - Categories, Algorithms, Programming (Gutsche, P., Skartsæterhagen)

CAP is a software project in GAP facilitating the implementation of

- 1 specific instances of categories,
- 2 category constructors,
- 3 categorical algorithms.

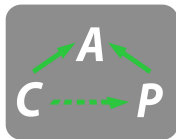




## CAP - Categories, Algorithms, Programming (Gutsche, P., Skartsæterhagen)

CAP is a software project in GAP facilitating the implementation of

- 1 specific instances of categories,
- 2 category constructors,
- 3 categorical algorithms.



## CAP - Categories, Algorithms, Programming (Gutsche, P., Skartsæterhagen)

CAP is a software project in GAP facilitating the implementation of

- 1 specific instances of categories,
- 2 [category constructors](#),
- 3 categorical algorithms.

# Category constructors



# Category constructors

category  $\longrightarrow$  category constructor  $\longrightarrow$  another category

## Example

# Category constructors

category  $\longrightarrow$  category constructor  $\longrightarrow$  another category

## Example

**A** additive  $\longrightarrow$  Freyd category  $\longrightarrow \mathcal{A}(\mathbf{A})$

# Category constructors

category  $\longrightarrow$  category constructor  $\longrightarrow$  another category

## Example

**A** additive  $\longrightarrow$  Freyd category  $\longrightarrow \mathcal{A}(\mathbf{A})$

# Freyd category

# Freyd category

## Freyd category: implicit characterization

The Freyd category of an additive category **A** is its universal cokernel completion



# Freyd category

## Freyd category: implicit characterization

The Freyd category of an additive category  $\mathbf{A}$  is its universal cokernel completion, i.e., it consists of

- a category  $\mathcal{A}(\mathbf{A})$  having cokernels
- a functor  $\mathbf{A} \rightarrow \mathcal{A}(\mathbf{A})$

# Freyd category

## Freyd category: implicit characterization

The Freyd category of an additive category  $\mathbf{A}$  is its universal cokernel completion, i.e., it consists of

- a category  $\mathcal{A}(\mathbf{A})$  having cokernels
- a functor  $\mathbf{A} \rightarrow \mathcal{A}(\mathbf{A})$

such that for all additive  $\mathbf{T}$  with cokernels and functors  $\mathbf{A} \rightarrow \mathbf{T}$ , we have:

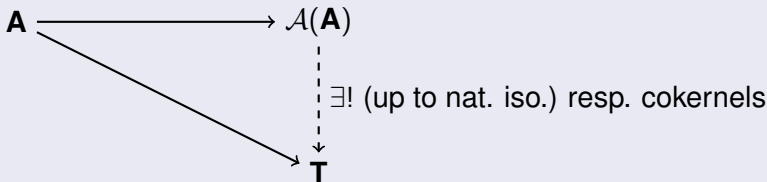
# Freyd category

## Freyd category: implicit characterization

The Freyd category of an additive category  $\mathbf{A}$  is its universal cokernel completion, i.e., it consists of

- a category  $\mathcal{A}(\mathbf{A})$  having cokernels
- a functor  $\mathbf{A} \rightarrow \mathcal{A}(\mathbf{A})$

such that for all additive  $\mathbf{T}$  with cokernels and functors  $\mathbf{A} \rightarrow \mathbf{T}$ , we have:



# Freyd category

# Freyd category

## Examples of Freyd categories

# Freyd category

## Examples of Freyd categories

- $\mathcal{A}(\text{Rows}_R) \simeq R\text{-fpmod}$

# Freyd category

## Examples of Freyd categories

- $\mathcal{A}(\text{Rows}_R) \simeq R\text{-fpmod}$
- $\mathcal{A}(\{ \text{Hom}(A, -) \mid A \in R\text{-fpmod} \}) \simeq$   
finitely presented functors over  $R\text{-fpmod}$

# Freyd category

## Examples of Freyd categories

- $\mathcal{A}(\text{Rows}_R) \simeq R\text{-fpmod}$
- $\mathcal{A}(\{ \text{Hom}(A, -) \mid A \in R\text{-fpmod} \}) \simeq$   
finitely presented functors over  $R\text{-fpmod}$

## Yoneda's lemma

$$\{ \text{Hom}(A, -) \mid A \in R\text{-fpmod} \} \simeq R\text{-fpmod}^{\text{op}}$$



# Freyd category

## Examples of Freyd categories

- $\mathcal{A}(\text{Rows}_R) \simeq R\text{-fpmod}$
- $\mathcal{A}(\{ \text{Hom}(A, -) \mid A \in R\text{-fpmod} \}) \simeq$   
finitely presented functors over  $R\text{-fpmod}$

## Yoneda's lemma

$$\{ \text{Hom}(A, -) \mid A \in R\text{-fpmod} \} \simeq R\text{-fpmod}^{\text{op}}$$

## Corollary

finitely presented functors over  $R\text{-fpmod} \simeq \mathcal{A}(\mathcal{A}(\text{Rows}_R)^{\text{op}})$

# Freyd category

## Examples of Freyd categories

- $\mathcal{A}(\text{Rows}_R) \simeq R\text{-fpmod}$
- $\mathcal{A}(\{ \text{Hom}(A, -) \mid A \in R\text{-fpmod} \}) \simeq$   
finitely presented functors over  $R\text{-fpmod}$

## Yoneda's lemma

$$\{ \text{Hom}(A, -) \mid A \in R\text{-fpmod} \} \simeq R\text{-fpmod}^{\text{op}}$$

## Corollary

finitely presented functors over  $R\text{-fpmod} \simeq \mathcal{A}(\mathcal{A}(\text{Rows}_R)^{\text{op}})$

**Study the constructiveness of  $\mathcal{A}(-)$ .**

# Freyd category

Let  $\mathbf{A}$  be additive.

# Freyd category

Let  $\mathbf{A}$  be additive.

Freyd category: data structures

The **Freyd category**  $\mathcal{A}(\mathbf{A})$  is given by the following data:

# Freyd category

Let  $\mathbf{A}$  be additive.

Freyd category: data structures

The **Freyd category**  $\mathcal{A}(\mathbf{A})$  is given by the following data:

- An object in  $\mathcal{A}(\mathbf{A})$  is simply a morphism  $(A \xleftarrow{\rho_A} R_A)$  in  $\mathbf{A}$ .

# Freyd category

Let  $\mathbf{A}$  be additive.

## Freyd category: data structures

The **Freyd category**  $\mathcal{A}(\mathbf{A})$  is given by the following data:

- An object in  $\mathcal{A}(\mathbf{A})$  is simply a morphism  $(A \xleftarrow{\rho_A} R_A)$  in  $\mathbf{A}$ .
- Morphisms:

# Freyd category

Let  $\mathbf{A}$  be additive.

## Freyd category: data structures

The **Freyd category**  $\mathcal{A}(\mathbf{A})$  is given by the following data:

- An object in  $\mathcal{A}(\mathbf{A})$  is simply a morphism  $(A \xleftarrow{\rho_A} R_A)$  in  $\mathbf{A}$ .

$$A \xleftarrow{\rho_A} R_A$$

- Morphisms:

# Freyd category

Let  $\mathbf{A}$  be additive.

## Freyd category: data structures

The **Freyd category**  $\mathcal{A}(\mathbf{A})$  is given by the following data:

- An object in  $\mathcal{A}(\mathbf{A})$  is simply a morphism  $(A \xleftarrow{\rho_A} R_A)$  in  $\mathbf{A}$ .

$$A \xleftarrow{\rho_A} R_A$$

- Morphisms:

$$B \xleftarrow{\rho_B} R_B$$



# Freyd category

Let  $\mathbf{A}$  be additive.

## Freyd category: data structures

The **Freyd category**  $\mathcal{A}(\mathbf{A})$  is given by the following data:

- An object in  $\mathcal{A}(\mathbf{A})$  is simply a morphism  $(A \xleftarrow{\rho_A} R_A)$  in  $\mathbf{A}$ .

- Morphisms:

$$\begin{array}{ccc} A & \xleftarrow{\rho_A} & R_A \\ \alpha \downarrow & & \\ B & \xleftarrow{\rho_B} & R_B \end{array}$$

# Freyd category

Let  $\mathbf{A}$  be additive.

## Freyd category: data structures

The **Freyd category**  $\mathcal{A}(\mathbf{A})$  is given by the following data:

- An object in  $\mathcal{A}(\mathbf{A})$  is simply a morphism  $(A \xleftarrow{\rho_A} R_A)$  in  $\mathbf{A}$ .

- Morphisms:

$$\begin{array}{ccc} A & \xleftarrow{\rho_A} & R_A \\ \alpha \downarrow & & \downarrow \exists \rho_\alpha \\ B & \xleftarrow{\rho_B} & R_B \end{array}$$

# Freyd category

Let  $\mathbf{A}$  be additive.

## Freyd category: data structures

The **Freyd category**  $\mathcal{A}(\mathbf{A})$  is given by the following data:

- An object in  $\mathcal{A}(\mathbf{A})$  is simply a morphism  $(A \xleftarrow{\rho_A} R_A)$  in  $\mathbf{A}$ .

- Morphisms:

$$\begin{array}{ccc} A & \xleftarrow{\rho_A} & R_A \\ \alpha \downarrow & \dashrightarrow & \downarrow \exists \rho_\alpha \\ B & \xleftarrow{\rho_B} & R_B \end{array} = 0$$

(A circle with a dot is located in the center of the square diagram.)

# Freyd category

Let  $\mathbf{A}$  be additive.

## Freyd category: data structures

The **Freyd category**  $\mathcal{A}(\mathbf{A})$  is given by the following data:

- An object in  $\mathcal{A}(\mathbf{A})$  is simply a morphism  $(A \xleftarrow{\rho_A} R_A)$  in  $\mathbf{A}$ .

- Morphisms:

$$\begin{array}{ccc}
 A & \xleftarrow{\rho_A} & R_A \\
 \alpha \downarrow & \dashrightarrow & \downarrow \exists \rho_\alpha \\
 B & \xleftarrow{\rho_B} & R_B
 \end{array}
 \quad = \quad 0$$

We also write

$$(A \xleftarrow{\rho_A} R_A) \xrightarrow{\alpha} (B \xleftarrow{\rho_B} R_B)$$

# Freyd category

Let  $\mathbf{A}$  be additive.

## Freyd category: data structures

The **Freyd category**  $\mathcal{A}(\mathbf{A})$  is given by the following data:

- An object in  $\mathcal{A}(\mathbf{A})$  is simply a morphism  $(A \xleftarrow{\rho_A} R_A)$  in  $\mathbf{A}$ .

- Morphisms:

$$\begin{array}{ccc}
 A & \xleftarrow{\rho_A} & R_A \\
 \downarrow \alpha & \searrow \text{dashed} & \downarrow \exists \rho_\alpha \\
 B & \xleftarrow{\rho_B} & R_B
 \end{array}
 \quad = \quad 0$$

We also write

$$(A \xleftarrow{\rho_A} R_A) \xrightarrow{\alpha} (B \xleftarrow{\rho_B} R_B)$$

# Algorithms in Freyd categories

Freyd category: algorithms

# Algorithms in Freyd categories

## Freyd category: algorithms

- **composition** in  $\mathbf{A}$   $\rightsquigarrow$  **composition** in  $\mathcal{A}(\mathbf{A})$

# Algorithms in Freyd categories

## Freyd category: algorithms

- **composition** in  $\mathbf{A}$   $\rightsquigarrow$  **composition** in  $\mathcal{A}(\mathbf{A})$
- **identities** in  $\mathbf{A}$   $\rightsquigarrow$  **identities** in  $\mathcal{A}(\mathbf{A})$



# Algorithms in Freyd categories

## Freyd category: algorithms

- **composition** in  $\mathbf{A}$   $\rightsquigarrow$  **composition** in  $\mathcal{A}(\mathbf{A})$
- **identities** in  $\mathbf{A}$   $\rightsquigarrow$  **identities** in  $\mathcal{A}(\mathbf{A})$
- **Cokernels** in  $\mathcal{A}(\mathbf{A})$

# Algorithms in Freyd categories

## Freyd category: algorithms

- **composition** in  $\mathbf{A}$   $\rightsquigarrow$  **composition** in  $\mathcal{A}(\mathbf{A})$
- **identities** in  $\mathbf{A}$   $\rightsquigarrow$  **identities** in  $\mathcal{A}(\mathbf{A})$
- **Cokernels** in  $\mathcal{A}(\mathbf{A})$

# Algorithms in Freyd categories

## Freyd category: algorithms

- **composition** in  $\mathbf{A}$   $\rightsquigarrow$  **composition** in  $\mathcal{A}(\mathbf{A})$
- **identities** in  $\mathbf{A}$   $\rightsquigarrow$  **identities** in  $\mathcal{A}(\mathbf{A})$
- **Cokernels** in  $\mathcal{A}(\mathbf{A})$

$$(A \xleftarrow{\rho_A} R_A) \xrightarrow{\alpha} (B \xleftarrow{\rho_B} R_B)$$

# Algorithms in Freyd categories

## Freyd category: algorithms

- **composition** in  $\mathbf{A}$   $\rightsquigarrow$  **composition** in  $\mathcal{A}(\mathbf{A})$
- **identities** in  $\mathbf{A}$   $\rightsquigarrow$  **identities** in  $\mathcal{A}(\mathbf{A})$
- **Cokernels** in  $\mathcal{A}(\mathbf{A})$

$$(B \xleftarrow{\begin{pmatrix} \rho_B \\ \alpha \end{pmatrix}} R_B \oplus A)$$

$$(A \xleftarrow{\rho_A} R_A) \xrightarrow{\alpha} (B \xleftarrow{\rho_B} R_B)$$

# Algorithms in Freyd categories

## Freyd category: algorithms

- **composition** in  $\mathbf{A}$   $\rightsquigarrow$  **composition** in  $\mathcal{A}(\mathbf{A})$
- **identities** in  $\mathbf{A}$   $\rightsquigarrow$  **identities** in  $\mathcal{A}(\mathbf{A})$
- **Cokernels** in  $\mathcal{A}(\mathbf{A})$

$$(A \xleftarrow{\rho_A} R_A) \xrightarrow{\alpha} (B \xleftarrow{\rho_B} R_B) \xrightarrow{\text{id}_B} (B \xleftarrow{\begin{pmatrix} \rho_B \\ \alpha \end{pmatrix}} R_B \oplus A)$$

# Algorithms in Freyd categories

## Freyd category: algorithms

- **composition** in  $\mathbf{A}$   $\rightsquigarrow$  **composition** in  $\mathcal{A}(\mathbf{A})$
- **identities** in  $\mathbf{A}$   $\rightsquigarrow$  **identities** in  $\mathcal{A}(\mathbf{A})$
- **Cokernels** in  $\mathcal{A}(\mathbf{A})$

A commutative diagram illustrating the construction of a cokernel in the Freyd category  $\mathcal{A}(\mathbf{A})$ . The diagram consists of three objects and three morphisms:

- Top-left object:  $(A \xleftarrow{\rho_A} R_A)$
- Top-right object:  $(B \xleftarrow{\rho_B} R_B \oplus A)$
- Bottom object:  $(T \xleftarrow{\rho_T} R_T)$

The morphisms are:

- A horizontal arrow from  $(A \xleftarrow{\rho_A} R_A)$  to  $(B \xleftarrow{\rho_B} R_B)$  labeled  $\alpha$ .
- A diagonal arrow from  $(B \xleftarrow{\rho_B} R_B)$  to  $(B \xleftarrow{\rho_B} R_B \oplus A)$  labeled  $\text{id}_B$ .
- A curved arrow from  $(A \xleftarrow{\rho_A} R_A)$  to  $(T \xleftarrow{\rho_T} R_T)$  labeled  $0$ .
- A diagonal arrow from  $(B \xleftarrow{\rho_B} R_B)$  to  $(T \xleftarrow{\rho_T} R_T)$  labeled  $\tau$ .

# Algorithms in Freyd categories

## Freyd category: algorithms

- **composition** in  $\mathbf{A} \rightsquigarrow$  **composition** in  $\mathcal{A}(\mathbf{A})$
- **identities** in  $\mathbf{A} \rightsquigarrow$  **identities** in  $\mathcal{A}(\mathbf{A})$
- **Cokernels** in  $\mathcal{A}(\mathbf{A})$

$$\begin{array}{ccc}
 & & (B \xleftarrow{\begin{pmatrix} \rho_B \\ \alpha \end{pmatrix}} R_B \oplus A) \\
 & \nearrow \text{id}_B & \\
 (A \xleftarrow{\rho_A} R_A) & \xrightarrow{\alpha} & (B \xleftarrow{\rho_B} R_B) \\
 & \searrow \tau & \\
 & & (T \xleftarrow{\rho_T} R_T)
 \end{array}$$

A curved arrow labeled  $0$  points from  $(A \xleftarrow{\rho_A} R_A)$  to  $(T \xleftarrow{\rho_T} R_T)$ .  
 A dashed arrow labeled  $\tau$  points from  $(B \xleftarrow{\begin{pmatrix} \rho_B \\ \alpha \end{pmatrix}} R_B \oplus A)$  to  $(T \xleftarrow{\rho_T} R_T)$ .

# Algorithms in Freyd categories

## Freyd category: algorithms

- **composition** in  $\mathbf{A} \rightsquigarrow$  **composition** in  $\mathcal{A}(\mathbf{A})$
- **identities** in  $\mathbf{A} \rightsquigarrow$  **identities** in  $\mathcal{A}(\mathbf{A})$
- **Cokernels** in  $\mathcal{A}(\mathbf{A})$  are constructive if  $\mathbf{A}$  is:

$$\begin{array}{ccc}
 & & (B \xleftarrow{\begin{pmatrix} \rho_B \\ \alpha \end{pmatrix}} R_B \oplus A) \\
 & \nearrow \text{id}_B & \\
 (A \xleftarrow{\rho_A} R_A) & \xrightarrow{\alpha} & (B \xleftarrow{\rho_B} R_B) \\
 & \searrow \tau & \\
 & & (T \xleftarrow{\rho_T} R_T)
 \end{array}$$

A curved arrow labeled  $0$  points from  $(A \xleftarrow{\rho_A} R_A)$  to  $(T \xleftarrow{\rho_T} R_T)$ .  
 A dashed arrow labeled  $\tau$  points from  $(B \xleftarrow{\begin{pmatrix} \rho_B \\ \alpha \end{pmatrix}} R_B \oplus A)$  to  $(T \xleftarrow{\rho_T} R_T)$ .



# Algorithms in Freyd categories

# Algorithms in Freyd categories

More delicate algorithmic issues:

More delicate algorithmic issues:

- **Kernels** in Freyd categories.

# Algorithms in Freyd categories

More delicate algorithmic issues:

- **Kernels** in Freyd categories.
- **Equality** of morphisms in Freyd categories.

# Algorithms in Freyd categories

More delicate algorithmic issues:

- **Kernels** in Freyd categories.
- **Equality** of morphisms in Freyd categories.

# Kernels in Freyd categories

## Theorem (Freyd)

Let  $\mathbf{A}$  be an additive category. Then  $\mathcal{A}(\mathbf{A})$  has kernels if and only if  $\mathbf{A}$  has weak kernels.

# Kernels in Freyd categories

## Theorem (Freyd)

Let  $\mathbf{A}$  be an additive category. Then  $\mathcal{A}(\mathbf{A})$  has kernels if and only if  $\mathbf{A}$  has weak kernels.

This theorem can be proven constructively. In particular, an algorithm for weak kernels in  $\mathbf{A}$  gives an algorithm for kernels in  $\mathcal{A}(\mathbf{A})$ .

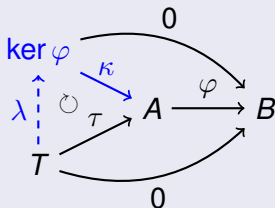




# Weak kernels

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

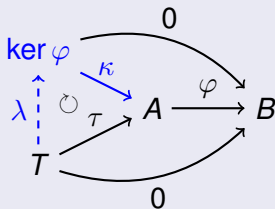
$\dots$  one needs an object  $\ker \varphi$ ,  
its embedding  $\kappa = \text{KernelEmbedding}(\varphi)$ ,  
and for every test morphism  $\tau$   
a *unique* morphism  $\lambda = \text{KernelLift}(\varphi, \tau)$ , such that



# Weak kernels

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

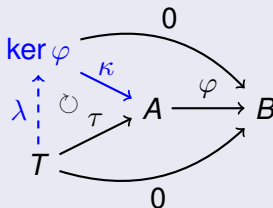
$\dots$  one needs an object  $\ker \varphi$ ,  
its embedding  $\kappa = \text{KernelEmbedding}(\varphi)$ ,  
and for every test morphism  $\tau$   
a *unique* morphism  $\lambda = \text{KernelLift}(\varphi, \tau)$ , such that



# Weak kernels

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

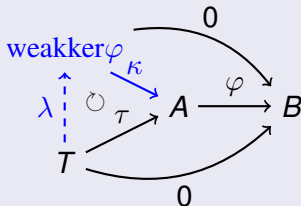
$\dots$  one needs an object  $\ker \varphi$ ,  
its embedding  $\kappa = \text{KernelEmbedding}(\varphi)$ ,  
and for every test morphism  $\tau$   
a *unique* morphism  $\lambda = \text{KernelLift}(\varphi, \tau)$ , such that



# Weak kernels

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

... one needs an object **weakker** $\varphi$ ,  
its embedding  $\kappa = \text{WeakKernelEmbedding}(\varphi)$ ,  
and for every test morphism  $\tau$   
a morphism  $\lambda = \text{WeakKernelLift}(\varphi, \tau)$ , such that



# Weak kernels in $\mathbf{Rows}_R$

# Weak kernels in $\text{Rows}_R$

## Weak kernels in $\text{Rows}_R$

$\text{Rows}_R$  has weak kernels iff the (row) kernel of every  $M \in R^{m \times n}$  is finitely generated.

# Weak kernels in $\text{Rows}_R$

## Weak kernels in $\text{Rows}_R$

$\text{Rows}_R$  has weak kernels iff the (row) kernel of every  $M \in R^{m \times n}$  is finitely generated. In this case  $R$  is called (left) **coherent**.

# Weak kernels in $\text{Rows}_R$

## Weak kernels in $\text{Rows}_R$

$\text{Rows}_R$  has weak kernels iff the (row) kernel of every  $M \in R^{m \times n}$  is finitely generated. In this case  $R$  is called (left) **coherent**.

## Remark

Algorithms computing such **syzygies** usually rely on Gröbner bases.



# Weak kernels in $\text{Rows}_R$

## Weak kernels in $\text{Rows}_R$

$\text{Rows}_R$  has weak kernels iff the (row) kernel of every  $M \in R^{m \times n}$  is finitely generated. In this case  $R$  is called (left) **coherent**.

## Remark

Algorithms computing such **syzygies** usually rely on Gröbner bases.

## Examples

- $k[x_1, \dots, x_n]$

# Weak kernels in $\text{Rows}_R$

## Weak kernels in $\text{Rows}_R$

$\text{Rows}_R$  has weak kernels iff the (row) kernel of every  $M \in R^{m \times n}$  is finitely generated. In this case  $R$  is called (left) **coherent**.

## Remark

Algorithms computing such **syzygies** usually rely on Gröbner bases.

## Examples

- $k[x_1, \dots, x_n]$  (coherent)

# Weak kernels in $\text{Rows}_R$

## Weak kernels in $\text{Rows}_R$

$\text{Rows}_R$  has weak kernels iff the (row) kernel of every  $M \in R^{m \times n}$  is finitely generated. In this case  $R$  is called (left) **coherent**.

## Remark

Algorithms computing such **syzygies** usually rely on Gröbner bases.

## Examples

- $k[x_1, \dots, x_n]$  (coherent)
- $k[x_i \mid i \in \mathbb{N}]$

# Weak kernels in $\text{Rows}_R$

## Weak kernels in $\text{Rows}_R$

$\text{Rows}_R$  has weak kernels iff the (row) kernel of every  $M \in R^{m \times n}$  is finitely generated. In this case  $R$  is called (left) **coherent**.

## Remark

Algorithms computing such **syzygies** usually rely on Gröbner bases.

## Examples

- $k[x_1, \dots, x_n]$  (coherent)
- $k[x_i \mid i \in \mathbb{N}]$  (coherent)

# Weak kernels in $\text{Rows}_R$

## Weak kernels in $\text{Rows}_R$

$\text{Rows}_R$  has weak kernels iff the (row) kernel of every  $M \in R^{m \times n}$  is finitely generated. In this case  $R$  is called (left) **coherent**.

## Remark

Algorithms computing such **syzygies** usually rely on Gröbner bases.

## Examples

- $k[x_1, \dots, x_n]$  (coherent)
- $k[x_i \mid i \in \mathbb{N}]$  (coherent)
- $k[z, x_i \mid i \in \mathbb{N}] / \langle zx_i \mid i \in \mathbb{N} \rangle$

# Weak kernels in $\text{Rows}_R$

## Weak kernels in $\text{Rows}_R$

$\text{Rows}_R$  has weak kernels iff the (row) kernel of every  $M \in R^{m \times n}$  is finitely generated. In this case  $R$  is called (left) **coherent**.

## Remark

Algorithms computing such **syzygies** usually rely on Gröbner bases.

## Examples

- $k[x_1, \dots, x_n]$  (coherent)
- $k[x_i \mid i \in \mathbb{N}]$  (coherent)
- $k[z, x_i \mid i \in \mathbb{N}] / \langle zx_i \mid i \in \mathbb{N} \rangle$  (not coherent)

# Weak kernels in $\mathcal{A}(\text{Rows}_R)^{\text{op}}$

# Weak kernels in $\mathcal{A}(\text{Rows}_R)^{\text{op}}$

We want to compute kernels in  $\mathcal{A}(\mathcal{A}(\text{Rows}_R)^{\text{op}})$ .



# Weak kernels in $\mathcal{A}(\text{Rows}_R)^{\text{op}}$

We want to compute kernels in  $\mathcal{A}(\mathcal{A}(\text{Rows}_R)^{\text{op}})$ .

- By Freyd's theorem, it suffices to have an algorithm for weak kernels in  $\mathcal{A}(\text{Rows}_R)^{\text{op}}$ .

# Weak kernels in $\mathcal{A}(\text{Rows}_R)^{\text{op}}$

We want to compute kernels in  $\mathcal{A}(\mathcal{A}(\text{Rows}_R)^{\text{op}})$ .

- By Freyd's theorem, it suffices to have an algorithm for weak kernels in  $\mathcal{A}(\text{Rows}_R)^{\text{op}}$ .
- Cokernels in  $\mathcal{A}(\text{Rows}_R)$  yield weak kernels in  $\mathcal{A}(\text{Rows}_R)^{\text{op}}$ .

# Weak kernels in $\mathcal{A}(\text{Rows}_R)^{\text{op}}$

We want to compute kernels in  $\mathcal{A}(\mathcal{A}(\text{Rows}_R)^{\text{op}})$ .

- By Freyd's theorem, it suffices to have an algorithm for weak kernels in  $\mathcal{A}(\text{Rows}_R)^{\text{op}}$ .
- Cokernels in  $\mathcal{A}(\text{Rows}_R)$  yield weak kernels in  $\mathcal{A}(\text{Rows}_R)^{\text{op}}$ .
- Cokernels in  $\mathcal{A}(\text{Rows}_R)$  are algorithmic.

# Weak kernels in $\mathcal{A}(\text{Rows}_R)^{\text{op}}$

We want to compute kernels in  $\mathcal{A}(\mathcal{A}(\text{Rows}_R)^{\text{op}})$ .

- By Freyd's theorem, it suffices to have an algorithm for weak kernels in  $\mathcal{A}(\text{Rows}_R)^{\text{op}}$ .
- Cokernels in  $\mathcal{A}(\text{Rows}_R)$  yield weak kernels in  $\mathcal{A}(\text{Rows}_R)^{\text{op}}$ .
- Cokernels in  $\mathcal{A}(\text{Rows}_R)$  are algorithmic.

$\rightsquigarrow$  Kernels in  $\mathcal{A}(\mathcal{A}(\text{Rows}_R)^{\text{op}})$  are algorithmic.

# Algorithms in Freyd categories

More delicate algorithmic issues:

- **Kernels** in Freyd categories.
- **Equality** of morphisms in Freyd categories.

# Algorithms in Freyd categories

More delicate algorithmic issues:

- **Kernels** in Freyd categories.
- **Equality** of morphisms in Freyd categories.

# Algorithms in Freyd categories

More delicate algorithmic issues:

- **Kernels** in Freyd categories. ✓
- **Equality** of morphisms in Freyd categories.

# Algorithms in Freyd categories

More delicate algorithmic issues:

- **Kernels** in Freyd categories. ✓
- **Equality** of morphisms in Freyd categories.



# Equality of morphisms in Freyd categories

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

# Equality of morphisms in Freyd categories

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

Freyd category: equality

Morphisms:

$$\begin{array}{ccc} A & \xleftarrow{\rho_A} & R_A \\ \alpha \downarrow & \searrow & \downarrow \exists \rho_\alpha \\ B & \xleftarrow{\rho_B} & R_B \end{array} = 0$$

# Equality of morphisms in Freyd categories

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

Freyd category: equality

Morphisms:

$$= 0$$

- $\mathbf{A} = \text{Rows}_R$ : linear system  $\textcolor{red}{X} \cdot \textcolor{blue}{D} = \textcolor{blue}{E}$  for matrices  $D, E$  in  $R$ .

# Equality of morphisms in Freyd categories

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

Freyd category: equality

Morphisms:

$$= 0$$

- $\mathbf{A} = \text{Rows}_R$ : linear system  $\textcolor{red}{X} \cdot \textcolor{blue}{D} = \textcolor{blue}{E}$  for matrices  $D, E$  in  $R$ .
- $\mathbf{A} = \mathcal{A}(\text{Rows}_R)^{\text{op}}$ : 2-sided linear system

# Equality of morphisms in Freyd categories

## Example

In  $k[x_1, \dots, x_n]$  we can solve all linear equations

# Equality of morphisms in Freyd categories

## Example

In  $k[x_1, \dots, x_n]$  we can solve all linear equations using Gröbner bases

# Equality of morphisms in Freyd categories

## Example

In  $k[x_1, \dots, x_n]/I$  we can solve all linear equations using Gröbner bases

# Equality of morphisms in Freyd categories

## Example

In  $S^{-1}k[x_1, \dots, x_n]/I$  we can solve all linear equations using Gröbner bases



# Equality of morphisms in Freyd categories

## Example

In  $S^{-1}k[x_1, \dots, x_n]/I$  we can solve all linear equations using Gröbner bases if we can also algorithmically create elements  $s \in S \cap J$  (if they exist) for any given ideal  $J \subseteq k[x_1, \dots, x_n]/I$ .

# Equality of morphisms in Freyd categories

## Example

In  $S^{-1}k[x_1, \dots, x_n]/I$  we can solve all linear equations using Gröbner bases if we can also algorithmically create elements  $s \in S \cap J$  (if they exist) for any given ideal  $J \subseteq k[x_1, \dots, x_n]/I$ .

## Nonexample

Let  $F$  be the free group in 10 generators.

# Equality of morphisms in Freyd categories

## Example

In  $S^{-1}k[x_1, \dots, x_n]/I$  we can solve all linear equations using Gröbner bases if we can also algorithmically create elements  $s \in S \cap J$  (if they exist) for any given ideal  $J \subseteq k[x_1, \dots, x_n]/I$ .

## Nonexample

Let  $F$  be the free group in 10 generators. In  $\mathbb{Q}[F \times F]$

# Equality of morphisms in Freyd categories

## Example

In  $S^{-1}k[x_1, \dots, x_n]/I$  we can solve all linear equations using Gröbner bases if we can also algorithmically create elements  $s \in S \cap J$  (if they exist) for any given ideal  $J \subseteq k[x_1, \dots, x_n]/I$ .

## Nonexample

Let  $F$  be the free group in 10 generators. In  $\mathbb{Q}[F \times F]$  the existence of a solution of a given linear system  $X \cdot D = E$  is **computationally undecidable**.

# Equality of morphisms in Freyd categories

## Example

In  $S^{-1}k[x_1, \dots, x_n]/I$  we can solve all linear equations using Gröbner bases if we can also algorithmically create elements  $s \in S \cap J$  (if they exist) for any given ideal  $J \subseteq k[x_1, \dots, x_n]/I$ .

## Nonexample

Let  $F$  be the free group in 10 generators. In  $\mathbb{Q}[F \times F]$  the existence of a solution of a given linear system  $X \cdot D = E$  is **computationally undecidable**. This is based on an example by Collins of a f.p. group with 10 generators with unsolvable word problem.

# Equality of morphisms in Freyd categories

Nonexample (P., arXiv:1712.03492)

We can construct a ring  $R$  with an algorithm for solving linear systems  
 $X \cdot D = E$

# Equality of morphisms in Freyd categories

## Nonexample (P., arXiv:1712.03492)

We can construct a ring  $R$  with an algorithm for solving linear systems  $X \cdot D = E$ , but the existence of a solution of a given 2-sided linear system is **computationally undecidable**.

# Equality of morphisms in Freyd categories

## Nonexample (P., arXiv:1712.03492)

We can construct a ring  $R$  with an algorithm for solving linear systems  $X \cdot D = E$ , but the existence of a solution of a given 2-sided linear system is **computationally undecidable**.



We can decide equality of morphisms in  $\mathcal{A}(\text{Rows}_R)$



# Equality of morphisms in Freyd categories

## Nonexample (P., arXiv:1712.03492)

We can construct a ring  $R$  with an algorithm for solving linear systems  $X \cdot D = E$ , but the existence of a solution of a given 2-sided linear system is **computationally undecidable**.



We can decide equality of morphisms in  $\mathcal{A}(\text{Rows}_R)$ , but not in  $\mathcal{A}(\mathcal{A}(\text{Rows}_R)^{\text{op}})$ .

# Algorithms in Freyd categories

More delicate algorithmic issues:

- **Kernels** in Freyd categories. ✓
- **Equality** of morphisms in Freyd categories.

# Algorithms in Freyd categories

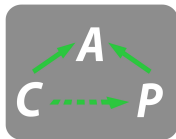
More delicate algorithmic issues:

- **Kernels** in Freyd categories. ✓
- **Equality** of morphisms in Freyd categories.

# Algorithms in Freyd categories

More delicate algorithmic issues:

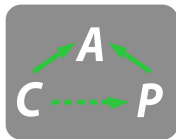
- **Kernels** in Freyd categories. ✓
- **Equality** of morphisms in Freyd categories. ✓



## CAP - Categories, Algorithms, Programming (Gutsche, P., Skartsæterhagen)

CAP is a software project in GAP facilitating the implementation of

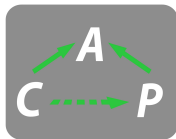
- 1 specific instances of categories,
- 2 [category constructors](#),
- 3 categorical algorithms.



## CAP - Categories, Algorithms, Programming (Gutsche, P., Skartsæterhagen)

CAP is a software project in GAP facilitating the implementation of

- 1 specific instances of categories,
- 2 category constructors,
- 3 categorical algorithms.



## CAP - Categories, Algorithms, Programming (Gutsche, P., Skartsæterhagen)

CAP is a software project in GAP facilitating the implementation of

- 1 specific instances of categories,
- 2 category constructors,
- 3 **categorical algorithms.**





## Categorical algorithms for abelian categories

## Categorical algorithms for abelian categories

- Abelian constructions like pullback or pushout

## Categorical algorithms for abelian categories

- Abelian constructions like pullback or pushout
- Constructive diagram chases

## Categorical algorithms for abelian categories

- Abelian constructions like pullback or pushout
- Constructive diagram chases
- Spectral sequence algorithm

# Categorical algorithms

## Categorical algorithms for abelian categories

- Abelian constructions like pullback or pushout
- Constructive diagram chases
- Spectral sequence algorithm

## Computational applications of Freyd categories

# Categorical algorithms

## Categorical algorithms for abelian categories

- Abelian constructions like pullback or pushout
- Constructive diagram chases
- Spectral sequence algorithm

## Computational applications of Freyd categories

- Computing sets of natural transformations, e.g.,  
 $\text{Hom}(\text{Tor}_j(M, -), \text{Ext}^i(A, -))$

# Categorical algorithms

## Categorical algorithms for abelian categories

- Abelian constructions like pullback or pushout
- Constructive diagram chases
- Spectral sequence algorithm

## Computational applications of Freyd categories

- Computing sets of natural transformations, e.g.,  
 $\text{Hom}(\text{Tor}_j(M, -), \text{Ext}^i(A, -))$
- Construct free abelian categories

# Categorical algorithms

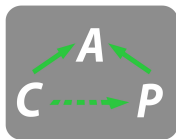
## Categorical algorithms for abelian categories

- Abelian constructions like pullback or pushout
- Constructive diagram chases
- Spectral sequence algorithm

## Computational applications of Freyd categories

- Computing sets of natural transformations, e.g.,  $\text{Hom}(\text{Tor}_j(M, -), \text{Ext}^i(A, -))$
- Construct free abelian categories  $\rightsquigarrow$  prove homological theorems

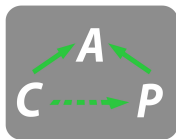




## CAP - Categories, Algorithms, Programming (Gutsche, P., Skartsæterhagen)

CAP is a software project in GAP facilitating the implementation of

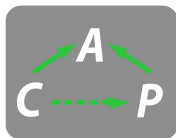
- 1 specific instances of categories,
- 2 category constructors,
- 3 **categorical algorithms.**



## CAP - Categories, Algorithms, Programming (Gutsche, P., Skartsæterhagen)

CAP is a software project in GAP facilitating the implementation of

- 1 specific instances of categories,
- 2 category constructors,
- 3 categorical algorithms.



## CAP - Categories, Algorithms, Programming (Gutsche, P., Skartsæterhagen)

CAP is a software project in GAP facilitating the implementation of

- 1 specific instances of categories,
- 2 category constructors,
- 3 categorical algorithms.

CAP Days 2018 in Siegen: 8/28/2018 - 8/31/2018

-  Maurice Auslander, *Coherent functors*, Proc. Conf. Categorical Algebra (La Jolla, Calif., 1965), Springer, New York, 1966, pp. 189–231. MR MR0212070 (35 #2945)
-  Mohamed Barakat and Markus Lange-Hegermann, *An axiomatic setup for algorithmic homological algebra and an alternative approach to localization*, J. Algebra Appl. **10** (2011), no. 2, 269–293, ([arXiv:1003.1943](#)). MR 2795737 (2012f:18022)
-  Peter Freyd, *Representations in abelian categories*, Proc. Conf. Categorical Algebra (La Jolla, Calif., 1965), Springer, New York, 1966, pp. 95–120. MR 0209333
-  Sebastian Posur, *A constructive approach to Freyd categories*, ArXiv e-prints (2017), ([arXiv:1712.03492](#)).