CAP - Categories, Algorithms, and Programming

Sebastian Gutsche and Sebastian Posur

TU Kaiserslautern, RWTH Aachen

September 28, 2015

What is CAP?

CAP means **Categories, algorithms, and programming**

What is CAP?

CAP means **Categories, algorithms, and programming** and is a software project implemented in GAP.

What is CAP?

CAP means **Categories, algorithms, and programming** and is a software project implemented in GAP.

- CAP derives powerful algorithms and data structures from basic categorical constructions.

What is CAP?

CAP means **Categories, algorithms, and programming** and is a software project implemented in GAP.

- CAP derives powerful algorithms and data structures from basic categorical constructions.
- CAP serves as a categorical programming language in which you can realize your code in a categorically structured way.

What is CAP?

CAP means **Categories, algorithms, and programming** and is a software project implemented in GAP.

- CAP derives powerful algorithms and data structures from basic categorical constructions.
- CAP serves as a categorical programming language in which you can realize your code in a categorically structured way.
- CAP simplifies complex computations by applying theorems.

What is CAP?

CAP means **Categories, algorithms, and programming** and is a software project implemented in GAP.

- CAP derives powerful algorithms and data structures from basic categorical constructions.
- CAP serves as a categorical programming language in which you can realize your code in a categorically structured way.
- CAP simplifies complex computations by applying theorems.

We call this concept **categorical programming**.

Outline

1 Motivation

Outline

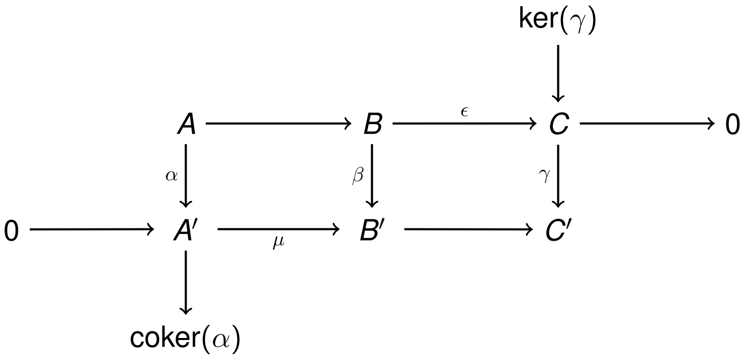
1 Motivation

2 Flavor of categorical programming in CAP

Section 1

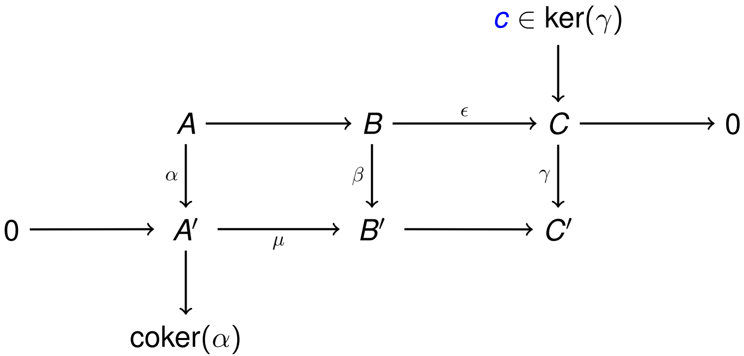
Motivation

Connecting Homomorphism in the Snake Lemma



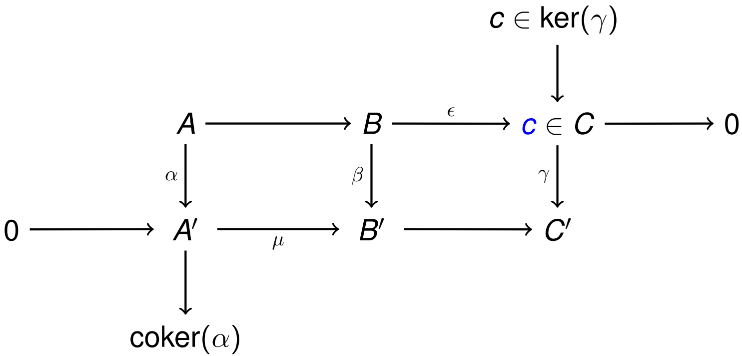
Wanted: ker(γ) $\xrightarrow{\partial}$ coker(α).

Connecting Homomorphism in the Snake Lemma



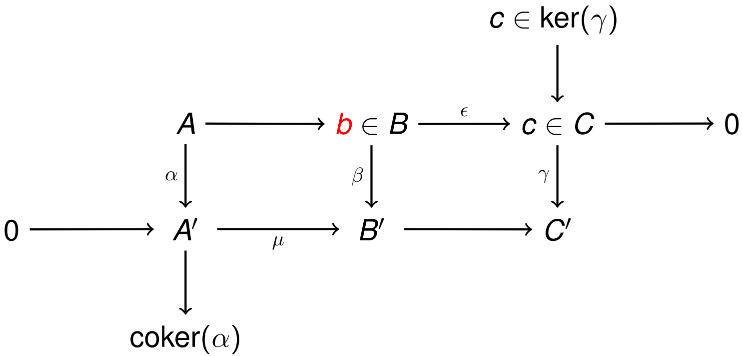
Start: $c \in \ker(\gamma)$.

Connecting Homomorphism in the Snake Lemma



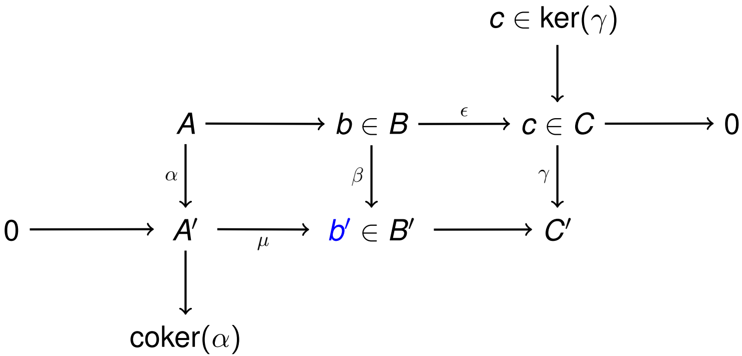
This lies in C.

Connecting Homomorphism in the Snake Lemma



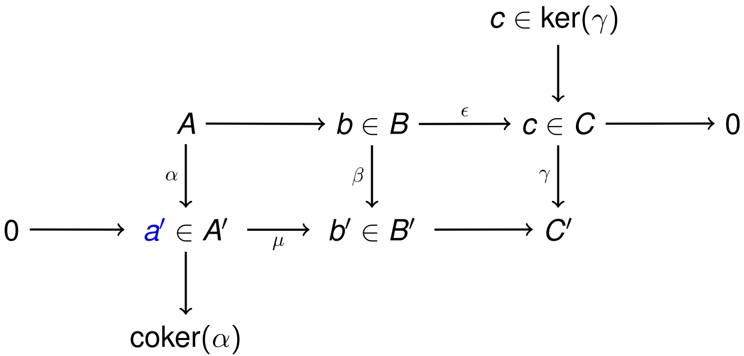
Choose: $b \in \epsilon^{-1}(\{c\})$.

Connecting Homomorphism in the Snake Lemma



Map: $b \overset{\beta}{\mapsto} b'$.

Connecting Homomorphism in the Snake Lemma



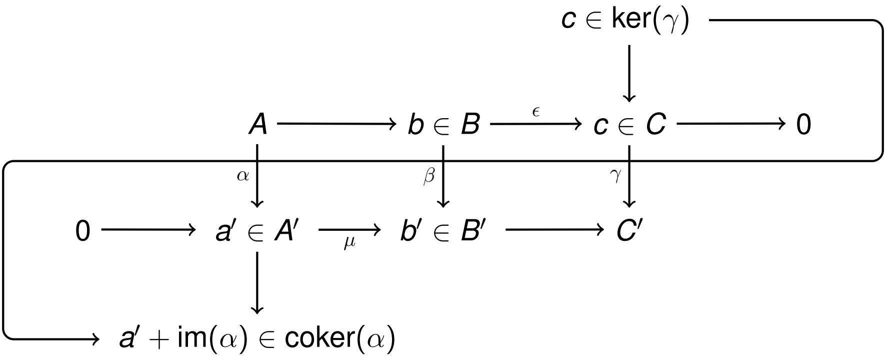
Compute: $a' \in \mu^{-1}(b')$.

Connecting Homomorphism in the Snake Lemma

$$
\begin{array}{ccccccccc}
 & & & & & & c \in \ker(\gamma) & & \\
 & & & & & & \downarrow & & \\
 & & A & \longrightarrow & b \in B & \xrightarrow{\epsilon} & c \in C & \longrightarrow & 0 \\
 & & \downarrow{\scriptstyle \alpha} & & \downarrow{\scriptstyle \beta} & & \downarrow{\scriptstyle \gamma} & & \\
0 & \longrightarrow & a' \in A' & \xrightarrow[\mu]{} & b' \in B' & \longrightarrow & C' & & \\
 & & \downarrow & & & & & & \\
 & & {\color{blue} a' + \mathrm{im}(\alpha)} \in \mathrm{coker}(\alpha) & & & & & &
\end{array}
$$

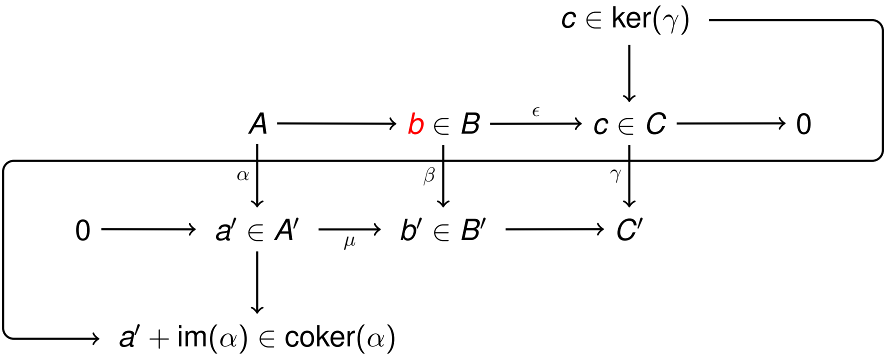
Map: $a' \mapsto a' + \mathrm{im}(\alpha)$.

Connecting Homomorphism in the Snake Lemma



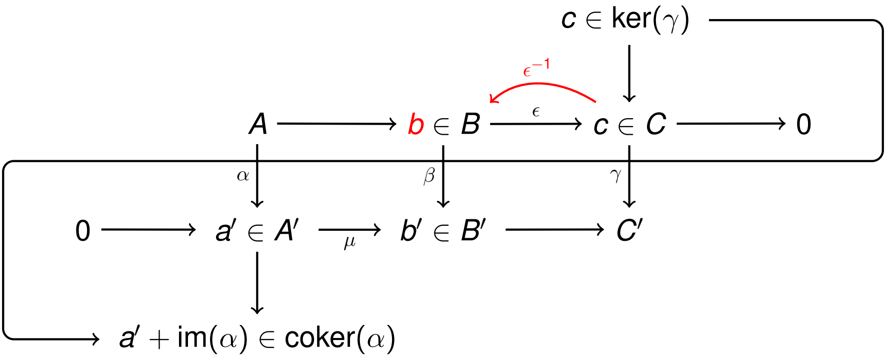
Result: $c \overset{\partial}{\mapsto} a' + \mathrm{im}(\alpha)$.

Connecting Homomorphism in the Snake Lemma



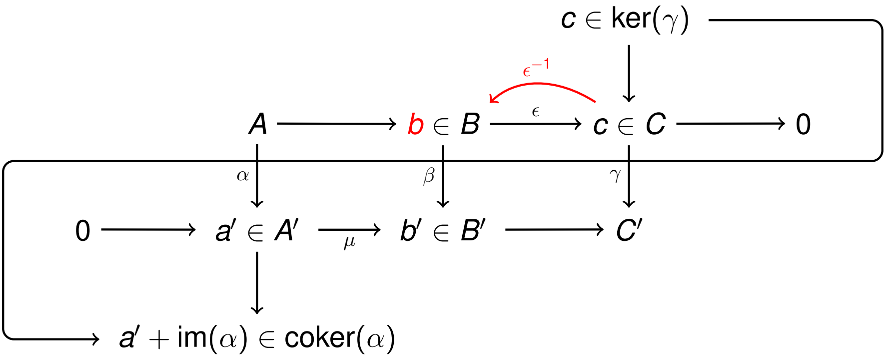
Result: $c \overset{\partial}{\mapsto} a' + \mathrm{im}(\alpha)$. Independent of the **choice**.

Connecting Homomorphism in the Snake Lemma



Any **right inverse** can be used.

Connecting Homomorphism in the Snake Lemma



Q: What if ϵ has no right inverse?

Generalized Morphism Category

Generalized Morphism Category

$\mathbf{A}$

Generalized Morphism Category

Generalized Morphism Category

$\mathbf{A}$ abelian category

Generalized Morphism Category

Generalized Morphism Category

$$\mathbf{A} \text{ abelian category} \xrightarrow{\text{CAP}}$$

Generalized Morphism Category

Generalized Morphism Category

$$\mathbf{A} \text{ abelian category} \xrightarrow{\textsc{Cap}} G(\mathbf{A})$$

Generalized Morphism Category

Generalized Morphism Category

$$\mathbf{A} \text{ abelian category} \xrightarrow{\textsc{Cap}} G(\mathbf{A})$$

Properties of $G(\mathbf{A})$

Generalized Morphism Category

Generalized Morphism Category

$$\mathbf{A} \text{ abelian category} \xrightarrow{\text{CAP}} G(\mathbf{A})$$

Properties of $G(\mathbf{A})$

- $\mathbf{A} \subseteq G(\mathbf{A})$.

Generalized Morphism Category

Generalized Morphism Category

$$\mathbf{A} \text{ abelian category} \xrightarrow{\text{CAP}} G(\mathbf{A})$$

Properties of $G(\mathbf{A})$

- $\mathbf{A} \subseteq G(\mathbf{A})$.
- Every monomorphism has a **canonical** left inverse.

Generalized Morphism Category

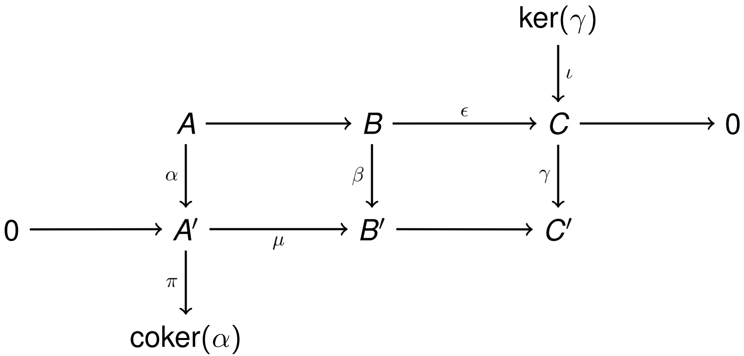
Generalized Morphism Category

$$\mathbf{A} \text{ abelian category} \xrightarrow{\text{CAP}} G(\mathbf{A})$$

Properties of $G(\mathbf{A})$

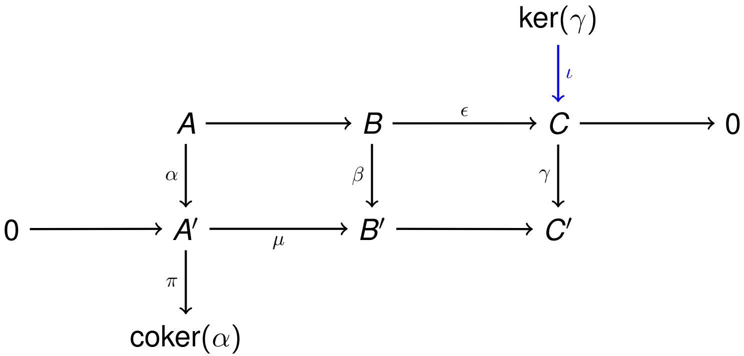
- $\mathbf{A} \subseteq G(\mathbf{A})$.
- Every monomorphism has a **canonical** left inverse.
- Every epimorphism has a **canonical** right inverse.

Snake Lemma Revisited



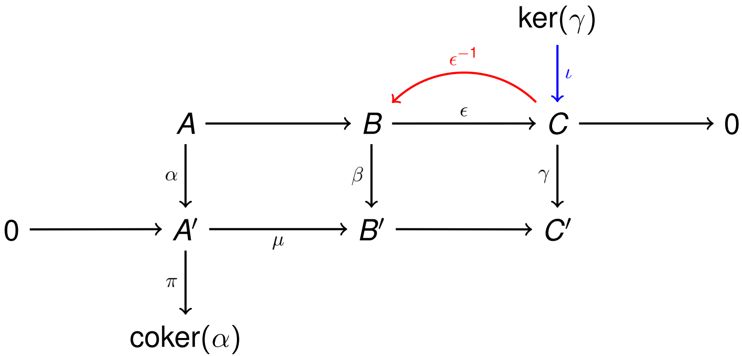
Wanted: $\ker(\gamma) \xrightarrow{\partial} \operatorname{coker}(\alpha)$.

Snake Lemma Revisited



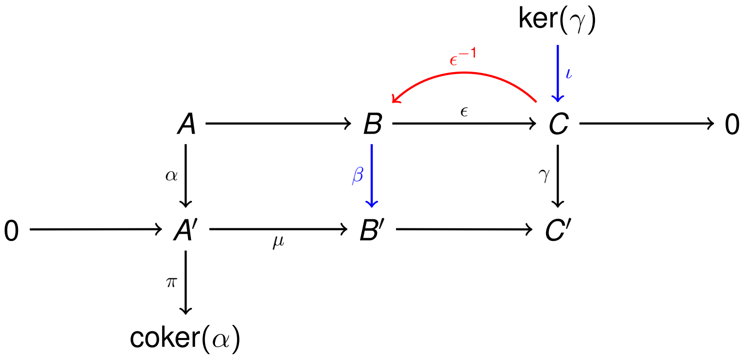
ι

Snake Lemma Revisited



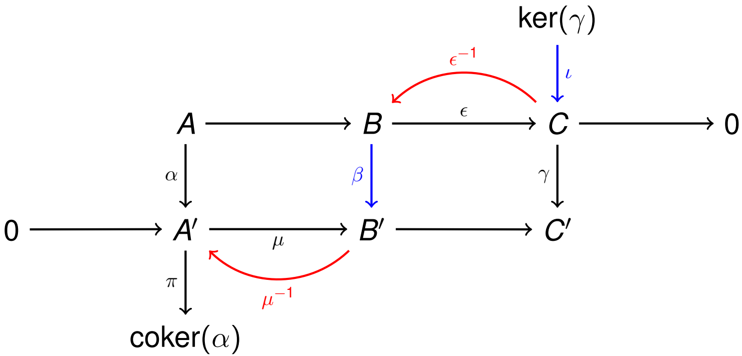
$$\epsilon^{-1} \circ \iota$$

Snake Lemma Revisited



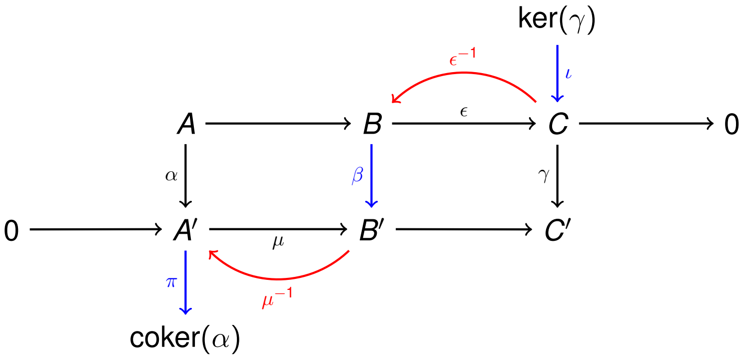
$$\beta \circ \epsilon^{-1} \circ \iota$$

Snake Lemma Revisited



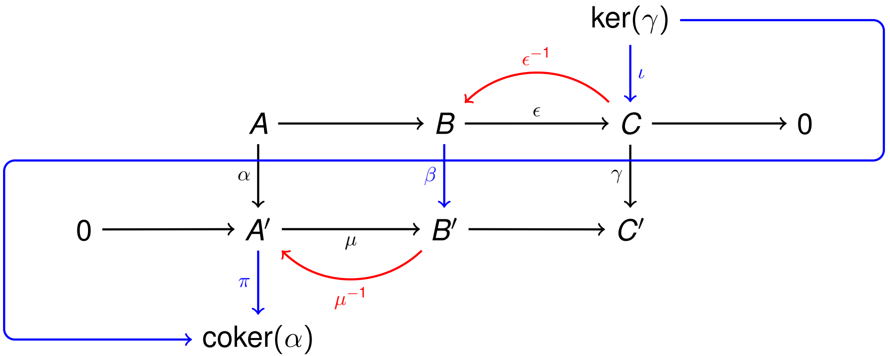
$$\mu^{-1} \circ \beta \circ \epsilon^{-1} \circ \iota$$

Snake Lemma Revisited



$$\pi \circ \mu^{-1} \circ \beta \circ \epsilon^{-1} \circ \iota$$

Snake Lemma Revisited



∂ is a composition of generalized morphisms!

Generalized Morphisms in CAP

Realization in CAP

- Generalized morphisms are implemented in CAP

Generalized Morphisms in CAP

Realization in CAP

- Generalized morphisms are implemented in CAP ($\mathbf{A} \mapsto G(\mathbf{A})$).

Generalized Morphisms in CAP

Realization in CAP

- Generalized morphisms are implemented in CAP ($\mathbf{A} \mapsto G(\mathbf{A})$).
- They are used for:

Generalized Morphisms in CAP

Realization in CAP

- Generalized morphisms are implemented in CAP ($\mathbf{A} \mapsto G(\mathbf{A})$).
- They are used for:
    1. Diagram chases

Generalized Morphisms in CAP

Realization in CAP

- Generalized morphisms are implemented in CAP ($\mathbf{A} \mapsto G(\mathbf{A})$).
- They are used for:
  1. Diagram chases
  2. Spectral sequences

Generalized Morphisms in CAP

Realization in CAP

- Generalized morphisms are implemented in CAP ($\mathbf{A} \mapsto G(\mathbf{A})$).
- They are used for:
  1. Diagram chases
  2. Spectral sequences
  3. Localization of categories

Generalized Morphisms in CAP

Realization in CAP

- Generalized morphisms are implemented in CAP ($\mathbf{A} \mapsto G(\mathbf{A})$).
- They are used for:
  1. Diagram chases
  2. Spectral sequences
  3. Localization of categories (Serre quotients)

Section 2

Flavor of categorical programming in CAP

Intersection of subobjects

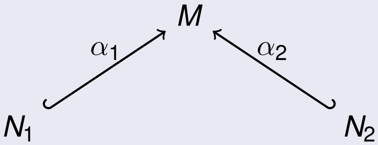
Let M be an object and $N_1 \hookrightarrow M$, $N_2 \hookrightarrow M$ subobjects.

Intersection of subobjects

Let M be an object and $N_1 \hookrightarrow M$, $N_2 \hookrightarrow M$ subobjects.
Task: Compute $N_1 \cap N_2 \hookrightarrow M$.

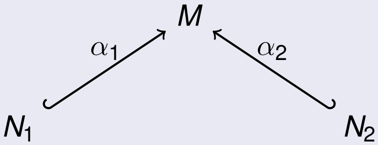
Intersection of subobjects

Let M be an object and $N_1 \hookrightarrow M$, $N_2 \hookrightarrow M$ subobjects.
Task: Compute $N_1 \cap N_2 \hookrightarrow M$.



Intersection of subobjects

Let M be an object and $N_1 \hookrightarrow M$, $N_2 \hookrightarrow M$ subobjects.
Task: Compute $N_1 \cap N_2 \hookrightarrow M$.

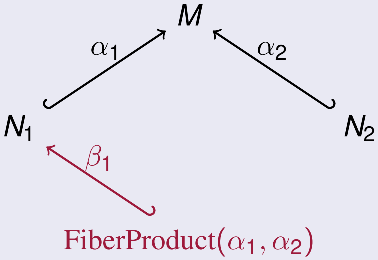


$\mathrm{FiberProduct}(\alpha_1, \alpha_2)$

1. Compute the fiber product of α_1 and α_2.

Intersection of subobjects

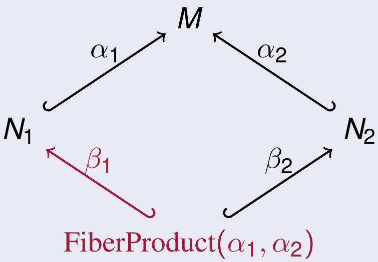
Let M be an object and $N_1 \hookrightarrow M$, $N_2 \hookrightarrow M$ subobjects.
Task: Compute $N_1 \cap N_2 \hookrightarrow M$.



1. Compute the fiber product of α_1 and α_2.
2. Compute the projection β_1.

Intersection of subobjects

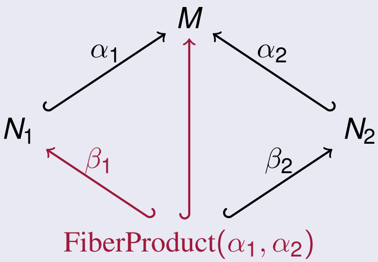
Let M be an object and $N_1 \hookrightarrow M$, $N_2 \hookrightarrow M$ subobjects.
Task: Compute $N_1 \cap N_2 \hookrightarrow M$.



1. Compute the fiber product of α_1 and α_2.
2. Compute the projection β_1.

Intersection of subobjects

Let M be an object and $N_1 \hookrightarrow M$, $N_2 \hookrightarrow M$ subobjects.
Task: Compute $N_1 \cap N_2 \hookrightarrow M$.



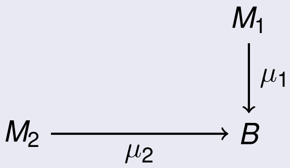
1. Compute the fiber product of α_1 and α_2.
2. Compute the projection β_1.
3. Return the composition $\alpha_1 \circ \beta_1$.

Fiber product computed by basic algorithms

Given $\mu_1 : M_1 \to B$ and $\mu_2 : M_2 \to B$,

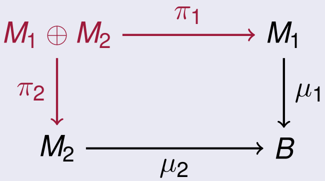
Fiber product computed by basic algorithms

Given $\mu_1 : M_1 \to B$ and $\mu_2 : M_2 \to B$, compute their fiber product.



Fiber product computed by basic algorithms

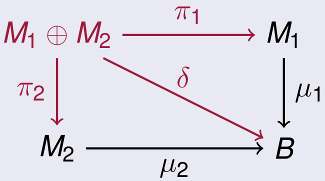
Given $\mu_1 : M_1 \to B$ and $\mu_2 : M_2 \to B$, compute their fiber product.



1. Compute $M_1 \oplus M_2$ and projection maps.

Fiber product computed by basic algorithms

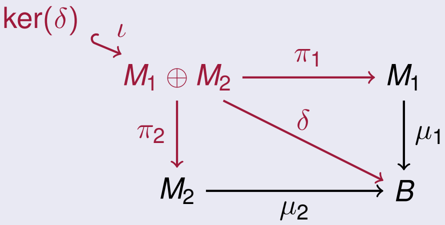
Given $\mu_1 : M_1 \to B$ and $\mu_2 : M_2 \to B$, compute their fiber product.



1. Compute $M_1 \oplus M_2$ and projection maps.
2. Compute $\delta := \mu_1 \circ \pi_1 - \mu_2 \circ \pi_2$.

Fiber product computed by basic algorithms

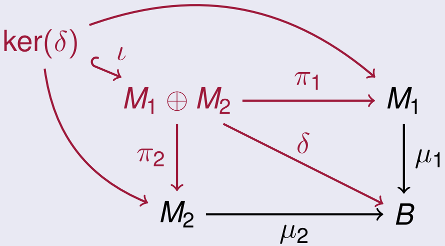
Given $\mu_1 : M_1 \to B$ and $\mu_2 : M_2 \to B$, compute their fiber product.



1. Compute $M_1 \oplus M_2$ and projection maps.
2. Compute $\delta := \mu_1 \circ \pi_1 - \mu_2 \circ \pi_2$.
3. Compute the kernel embedding ι of δ

Fiber product computed by basic algorithms

Given $\mu_1 : M_1 \to B$ and $\mu_2 : M_2 \to B$, compute their fiber product.



1. Compute $M_1 \oplus M_2$ and projection maps.
2. Compute $\delta := \mu_1 \circ \pi_1 - \mu_2 \circ \pi_2$.
3. Compute the kernel embedding ι of δ, $\pi_1 \circ \iota$, and $\pi_2 \circ \iota$.

Kernel

Let φ be a morphism in an additive category.

Kernel

Let φ be a morphism in an additive category. To handle the kernel of φ algorithmically . . .

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

Kernel

Let φ be a morphism in an additive category. To handle the kernel of φ algorithmically ...

... one has to construct the object $\ker\varphi$,

$\ker\varphi$

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

Kernel

Let φ be a morphism in an additive category. To handle the kernel of φ algorithmically . . .

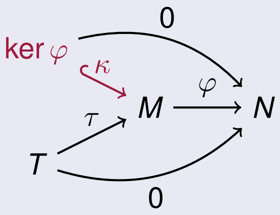
. . . one has to construct the object kerφ,
its embedding into the object M,

$$\ker\varphi \xhookrightarrow{\kappa} M \xrightarrow{\varphi} N$$

Kernel

Let φ be a morphism in an additive category. To handle the kernel of φ algorithmically . . .

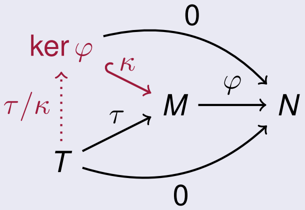
. . . one has to construct the object kerφ,
its embedding into the object M,
and for every test object T



Kernel

Let φ be a morphism in an additive category. To handle the kernel of φ algorithmically ...

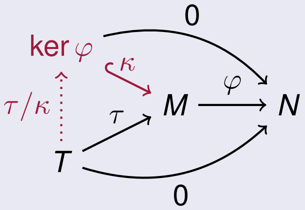
... one has to construct the object kerφ,
its embedding into the object M,
and for every test object T
a morphism given by kerφ's universal property.



Kernel

Let φ be a morphism in an additive category. To handle the kernel of φ algorithmically . . .

. . . one has to construct the object kerφ,
its embedding into the object M,
and for every test object T
a morphism given by kerφ's universal property.



Thus a proper implementation of the kernel needs **three** algorithms.

Basic operations

Build up your algorithms from basic categorical operations:

Basic operations

Build up your algorithms from basic categorical operations:

Example: Basic operations for fiber product

Basic operations

Build up your algorithms from basic categorical operations:

Example: Basic operations for fiber product

- Direct sum and projections in summands

Basic operations

Build up your algorithms from basic categorical operations:

Example: Basic operations for fiber product

- Direct sum and projections in summands
- Composition and subtraction of morphisms

Basic operations

Build up your algorithms from basic categorical operations:

Example: Basic operations for fiber product

- Direct sum and projections in summands
- Composition and subtraction of morphisms
- Kernel with embedding

Basic operations

Build up your algorithms from basic categorical operations:

Example: Basic operations for fiber product

- Direct sum and projections in summands
- Composition and subtraction of morphisms
- Kernel with embedding
- Universal property of kernel

Basic operations

Build up your algorithms from basic categorical operations:

Example: Basic operations for fiber product

- Direct sum and projections in summands
- Composition and subtraction of morphisms
- Kernel with embedding
- Universal property of kernel

Basic operations trigger different algorithms, depending on the context:

Basic operations

Build up your algorithms from basic categorical operations:

Example: Basic operations for fiber product

- Direct sum and projections in summands
- Composition and subtraction of morphisms
- Kernel with embedding
- Universal property of kernel

Basic operations trigger different algorithms, depending on the context:

Example: Kernel

Basic operations

Build up your algorithms from basic categorical operations:

Example: Basic operations for fiber product

- Direct sum and projections in summands
- Composition and subtraction of morphisms
- Kernel with embedding
- Universal property of kernel

Basic operations trigger different algorithms, depending on the context:

Example: Kernel

- Vector spaces $\vdash$ Gaussian elimination

Basic operations

Build up your algorithms from basic categorical operations:

Example: Basic operations for fiber product

- Direct sum and projections in summands
- Composition and subtraction of morphisms
- Kernel with embedding
- Universal property of kernel

Basic operations trigger different algorithms, depending on the context:

Example: Kernel

- Vector spaces $\vdash$ Gaussian elimination
- Modules $\vdash$ Gröbner basis computation

Available Constructions

Once all basic operations for a category **A** are implemented,

Available Constructions

Once all basic operations for a category **A** are implemented, CAP can (among other) create

Available Constructions

Once all basic operations for a category **A** are implemented, CAP can (among other) create

- $G(\mathbf{A})$, the generalized morphism category of **A**,

Available Constructions

Once all basic operations for a category **A** are implemented, CAP can (among other) create

- $G(\mathbf{A})$, the generalized morphism category of **A**,
- Serre quotient categories of **A**,

Available Constructions

Once all basic operations for a category **A** are implemented, CAP can (among other) create

- $G(\mathbf{A})$, the generalized morphism category of **A**,
- Serre quotient categories of **A**,
- the categories of complexes and cocomplexes of **A**,

Available Constructions

Once all basic operations for a category $\mathbf{A}$ are implemented, CAP can (among other) create

- $G(\mathbf{A})$, the generalized morphism category of $\mathbf{A}$,
- Serre quotient categories of $\mathbf{A}$,
- the categories of complexes and cocomplexes of $\mathbf{A}$,
- the categories of filtered objects of $\mathbf{A}$

Available Constructions

Once all basic operations for a category **A** are implemented, CAP can (among other) create

- $G(\mathbf{A})$, the generalized morphism category of **A**,
- Serre quotient categories of **A**,
- the categories of complexes and cocomplexes of **A**,
- the categories of filtered objects of **A**
- and combinations of those!

Available Algorithms

Using the basic operations and the constructions described, CAP can compute

Available Algorithms

Using the basic operations and the constructions described, CAP can compute

- spectral sequences,

Available Algorithms

Using the basic operations and the constructions described, CAP can compute

- spectral sequences,
- diagram chases,

Available Algorithms

Using the basic operations and the constructions described, CAP can compute

- spectral sequences,
- diagram chases,
- natural isomorphisms,

Available Algorithms

Using the basic operations and the constructions described, CAP can compute

- spectral sequences,
- diagram chases,
- natural isomorphisms,
- and much more!

Stacking of data structures

Example: Snake lemma in coherent sheaves over $\mathbb{P}^n$

Stacking of data structures

Example: Snake lemma in coherent sheaves over $\mathbb{P}^n$

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

Stacking of data structures

Example: Snake lemma in coherent sheaves over $\mathbb{P}^n$

$$G\left(\mathfrak{Coh}\left(\mathbb{P}^n\right)\right)$$
$$\downarrow$$
$$\mathfrak{Coh}\left(\mathbb{P}^n\right) \cong S - \mathrm{grmod} / S - \mathrm{grmod}^0$$

Stacking of data structures

Example: Snake lemma in coherent sheaves over $\mathbb{P}^n$

$$
\begin{array}{c}
G\left(\mathfrak{Coh}\left(\mathbb{P}^n\right)\right) \\
\downarrow \\
\mathfrak{Coh}\left(\mathbb{P}^n\right) \cong S-\mathrm{grmod}\,/\,S-\mathrm{grmod}^0 \\
\downarrow \\
G\left(S-\mathrm{grmod}\right)
\end{array}
$$

Stacking of data structures

Example: Snake lemma in coherent sheaves over $\mathbb{P}^n$

$$G\left(\mathfrak{Coh}\left(\mathbb{P}^n\right)\right)$$
$$\downarrow$$
$$\mathfrak{Coh}\left(\mathbb{P}^n\right) \cong S - \mathrm{grmod} / S - \mathrm{grmod}^0$$
$$\downarrow$$
$$G\left(S - \mathrm{grmod}\right)$$
$$\downarrow$$
$$S - \mathrm{grmod}$$

Stacking of data structures

Example: Snake lemma in coherent sheaves over $\mathbb{P}^n$

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

$$\downarrow$$

$$\mathfrak{Coh}\left(\mathbb{P}^n\right) \cong S-\text{grmod}\,/\,S-\text{grmod}^0$$

$$\downarrow$$

$$G\left(S-\text{grmod}\right)$$

$$\downarrow$$

$$S-\text{grmod}$$

$$\downarrow$$

$$\text{Projectives}(S)$$

Download CAP

CAP is currently not deposited with GAP. You can download it from GitHub:

`https://github.com/homalg-project/CAP_project`