Category theory as a foundation for algorithms and programming in computer algebra

Sebastian Posur

RWTH Aachen

January 15, 2016

Outline

1. Categories, Algorithms, and Programming

Outline

1. Categories, Algorithms, and Programming
2. Generalized Morphisms

Section 1

Categories, Algorithms, and Programming

Spectral Sequence Algorithm

Let **A** be an abelian category (e.g. the category of abelian groups).

Spectral Sequence Algorithm

Let **A** be an abelian category (e.g. the category of abelian groups).

Q: How to handle spectral sequences in **A** algorithmically?

Spectral Sequence Algorithm

Let **A** be an abelian category (e.g. the category of abelian groups).

Q: How to handle spectral sequences in **A** algorithmically?

Spectral Sequence Algorithm

$$\boxed{\mathcal{F} \in \mathrm{Filt}\left(\mathrm{Ch}\left(\mathbf{A}\right)\right)}$$

Spectral Sequence Algorithm

Let **A** be an abelian category (e.g. the category of abelian groups).

Q: How to handle spectral sequences in **A** algorithmically?

Spectral Sequence Algorithm

$\boxed{\mathcal{F} \in \mathrm{Filt}\left(\mathrm{Ch}\left(\mathbf{A}\right)\right)}$

3-dimensional array of objects and morphisms

Spectral Sequence Algorithm

Let **A** be an abelian category (e.g. the category of abelian groups).

Q: How to handle spectral sequences in **A** algorithmically?

Spectral Sequence Algorithm

$$\boxed{\mathcal{F} \in \mathrm{Filt}(\mathrm{Ch}(\mathbf{A}))} \xrightarrow{\text{Spectral Sequence Algorithm}} \boxed{\begin{array}{c}\text{3-dimensional}\\ \text{array of objects}\\ \text{and morphisms}\end{array}}$$

Spectral Sequence Algorithm

Let **A** be an abelian category (e.g. the category of abelian groups).

Q: How to handle spectral sequences in **A** algorithmically?

Spectral Sequence Algorithm

$$\boxed{\mathcal{F} \in \mathrm{Filt}\left(\mathrm{Ch}\left(\mathbf{A}\right)\right)} \xrightarrow{\text{Spectral Sequence Algorithm}} \boxed{\begin{array}{c}\text{3-dimensional}\\ \text{array of objects}\\ \text{and morphisms}\end{array}}$$

Depends only on primitive operations dictated by the axioms.

Spectral Sequence Algorithm

Let **A** be an abelian category (e.g. the category of abelian groups).

Q: How to handle spectral sequences in **A** algorithmically?

Spectral Sequence Algorithm

$$\boxed{\mathcal{F} \in \mathrm{Filt}\left(\mathrm{Ch}\left(\mathbf{A}\right)\right)} \xrightarrow{\text{Spectral Sequence Algorithm}} \boxed{\begin{array}{c}\text{3-dimensional}\\ \text{array of objects}\\ \text{and morphisms}\end{array}}$$

Depends only on primitive operations dictated by the axioms.

Axioms of on Abelian Category

Needed operations: $\circ, +, -$

Axioms of on Abelian Category

Needed operations: $\circ, +, -, \oplus, \mathrm{Ker}, \mathrm{Coker}$.

Axioms of on Abelian Category

Needed operations: $\circ, +, -, \oplus, \mathrm{Ker}, \mathrm{Coker}$.

Axioms of on Abelian Category

Needed operations: $\circ, +, -, \oplus$, Ker, Coker. Let φ be a morphism.

Axioms of on Abelian Category

Needed operations: $\circ, +, -, \oplus, \mathrm{Ker}, \mathrm{Coker}$. Let φ be a morphism. To handle the kernel of φ algorithmically . . .

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

Axioms of on Abelian Category

Needed operations: $\circ, +, -, \oplus, \mathrm{Ker}, \mathrm{Coker}$. Let φ be a morphism. To handle the kernel of φ algorithmically …

… one has to construct the object kerφ,

$$\ker\varphi$$

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

Axioms of on Abelian Category

Needed operations: $\circ, +, -, \oplus, \mathrm{Ker}, \mathrm{Coker}$. Let φ be a morphism. To handle the kernel of φ algorithmically ...

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

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

Axioms of on Abelian Category

Needed operations: $\circ, +, -, \oplus$, Ker, Coker. Let φ be a morphism. To handle the kernel of φ algorithmically ...

... one has to construct the object $\ker\varphi$,
its embedding into the object M,
and for every test object T

$$
\begin{tikzcd}
\ker\varphi \arrow[dr, hook, "\kappa"] \arrow[drr, bend left, "0"] & & \\
& M \arrow[r, "\varphi"] & N \\
T \arrow[ur, "\tau"] \arrow[urr, bend right, "0"'] & &
\end{tikzcd}
$$

Axioms of on Abelian Category

Needed operations: $\circ, +, -, \oplus$, Ker, Coker. Let φ be a morphism. 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.

$$\begin{tikzcd}
\ker\varphi \arrow[dr, hook, "\kappa"] \arrow[drr, bend left, "0"] & & \\
 & M \arrow[r, "\varphi"] & N \\
T \arrow[uu, dotted, "\tau/\kappa"] \arrow[ur, "\tau"] \arrow[urr, bend right, "0"'] & &
\end{tikzcd}$$

Axioms of on Abelian Category

Needed operations: $\circ, +, -, \oplus$, Ker, Coker. Let φ be a morphism. To handle the kernel of φ algorithmically ...

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

$$
\begin{tikzcd}
\ker\varphi \arrow[dr, hook, "\kappa"] \arrow[drr, bend left, "0"] & & \\
& M \arrow[r, "\varphi"] & N \\
T \arrow[uu, dotted, "\tau/\kappa"] \arrow[ur, "\tau"] \arrow[urr, bend right, "0"'] & &
\end{tikzcd}
$$

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

CAP

So we need a computer algebra system with . . .

CAP

So we need a computer algebra system with . . .

- the possibility to implement sophisticated categorical algorithms and data structures using primitive categorical operations,

CAP

So we need a computer algebra system with . . .

- the possibility to implement sophisticated categorical algorithms and data structures using primitive categorical operations,
- an interface to interpret these primitive categorical operations by computable operations (of various computer algebra systems).

CAP

So we need a computer algebra system with …

- the possibility to implement sophisticated categorical algorithms and data structures using primitive categorical operations,
- an interface to interpret these primitive categorical operations by computable operations (of various computer algebra systems).

$\rightsquigarrow$ CAP

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 (Groups, algorithms, and programming)

What is CAP?

CAP means **Categories, algorithms, and programming** and is a software project implemented in GAP (Groups, algorithms, and programming) (joint work with Sebastian Gutsche, based on ideas of the `homalg` project (Mohamed Barakat).

What is CAP?

CAP means **Categories, algorithms, and programming** and is a software project implemented in GAP (Groups, algorithms, and programming) (joint work with Sebastian Gutsche, based on ideas of the `homalg` project (Mohamed Barakat).

- CAP serves as a categorical programming language with categorical operations as primitives.

What is CAP?

CAP means **Categories, algorithms, and programming** and is a software project implemented in GAP (Groups, algorithms, and programming) (joint work with Sebastian Gutsche, based on ideas of the `homalg` project (Mohamed Barakat).

- CAP serves as a categorical programming language with categorical operations as primitives.
- CAP has an interface for the interpretation of these categorical operations.

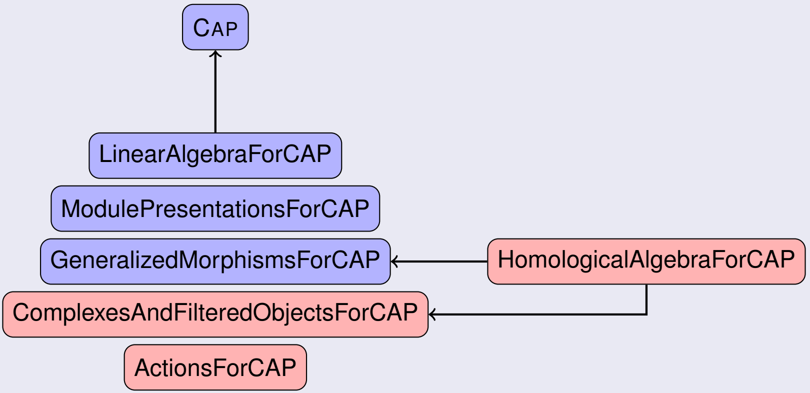
What is CAP?

CAP means **Categories, algorithms, and programming** and is a software project implemented in GAP (Groups, algorithms, and programming) (joint work with Sebastian Gutsche, based on ideas of the `homalg` project (Mohamed Barakat).

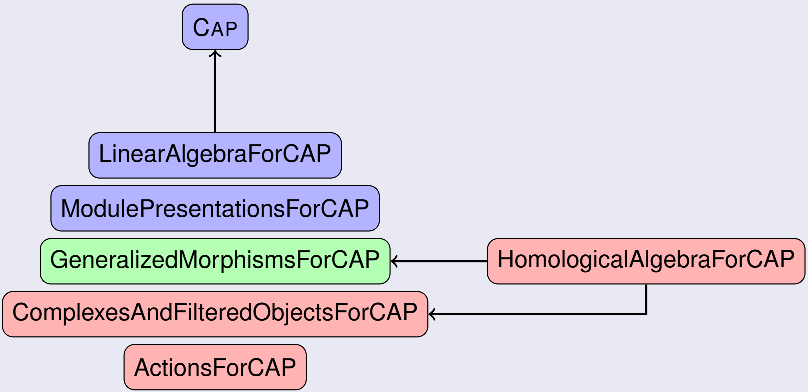
- CAP serves as a categorical programming language with categorical operations as primitives.
- CAP has an interface for the interpretation of these categorical operations.

Using CAP we are now able to implement the categorical data structures and algorithms which we need.

CAP Packages



CAP Packages



Section 2

Generalized Morphisms

Connecting Homomorphism in the Snake Lemma

$$\begin{tikzcd}
 & & & \ker(\gamma) \arrow[d] & \\
 & A \arrow[r] \arrow[d, "\alpha"'] & B \arrow[r, "\epsilon"] \arrow[d, "\beta"'] & C \arrow[r] \arrow[d, "\gamma"'] & 0 \\
0 \arrow[r] & A' \arrow[r, "\mu"'] \arrow[d] & B' \arrow[r] & C' & \\
 & \mathrm{coker}(\alpha) & & &
\end{tikzcd}$$

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

Connecting Homomorphism in the Snake Lemma

$$\begin{tikzcd}
 & & & {\color{blue}c} \in \ker(\gamma) \arrow[d] & \\
 & A \arrow[r] \arrow[d, "\alpha"'] & B \arrow[r, "\epsilon"] \arrow[d, "\beta"'] & C \arrow[r] \arrow[d, "\gamma"'] & 0 \\
0 \arrow[r] & A' \arrow[r, "\mu"'] \arrow[d] & B' \arrow[r] & C' & \\
 & \mathrm{coker}(\alpha) & & &
\end{tikzcd}$$

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

Connecting Homomorphism in the Snake Lemma

$$\begin{tikzcd}
 & & & c \in \ker(\gamma) \arrow[d] & \\
 & A \arrow[r] \arrow[d, "\alpha"'] & B \arrow[r, "\epsilon"] \arrow[d, "\beta"'] & {\color{blue}c} \in C \arrow[r] \arrow[d, "\gamma"'] & 0 \\
0 \arrow[r] & A' \arrow[r, "\mu"'] \arrow[d] & B' \arrow[r] & C' & \\
 & \operatorname{coker}(\alpha) & & &
\end{tikzcd}$$

This lies in C.

Connecting Homomorphism in the Snake Lemma

$$
\begin{tikzcd}
 & & & c \in \ker(\gamma) \arrow[d] & \\
 & A \arrow[r] \arrow[d, "\alpha"'] & \textcolor{red}{b} \in B \arrow[r, "\epsilon"] \arrow[d, "\beta"'] & c \in C \arrow[r] \arrow[d, "\gamma"'] & 0 \\
0 \arrow[r] & A' \arrow[r, "\mu"'] \arrow[d] & B' \arrow[r] & C' & \\
 & \operatorname{coker}(\alpha) & & &
\end{tikzcd}
$$

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

Connecting Homomorphism in the Snake Lemma

$$\begin{tikzcd}
 & & & c \in \ker(\gamma) \arrow[d] & \\
 & A \arrow[r] \arrow[d, "\alpha"'] & b \in B \arrow[r, "\epsilon"] \arrow[d, "\beta"'] & c \in C \arrow[r] \arrow[d, "\gamma"'] & 0 \\
0 \arrow[r] & A' \arrow[r, "\mu"'] \arrow[d] & {\color{blue}b'} \in B' \arrow[r] & C' & \\
 & \mathrm{coker}(\alpha) & & &
\end{tikzcd}$$

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

Connecting Homomorphism in the Snake Lemma

$$
\begin{tikzcd}
 & & & c \in \ker(\gamma) \arrow[d] & \\
 & A \arrow[r] \arrow[d, "\alpha"'] & b \in B \arrow[r, "\epsilon"] \arrow[d, "\beta"'] & c \in C \arrow[r] \arrow[d, "\gamma"'] & 0 \\
0 \arrow[r] & {\color{blue}a'} \in A' \arrow[r, "\mu"'] \arrow[d] & b' \in B' \arrow[r] & C' & \\
 & \mathrm{coker}(\alpha) & & &
\end{tikzcd}
$$

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

Connecting Homomorphism in the Snake Lemma

$$\begin{tikzcd}
& & & c \in \ker(\gamma) \arrow[d] & \\
& A \arrow[r] \arrow[d, "\alpha"'] & b \in B \arrow[r, "\epsilon"] \arrow[d, "\beta"'] & c \in C \arrow[r] \arrow[d, "\gamma"'] & 0 \\
0 \arrow[r] & a' \in A' \arrow[r, "\mu"'] \arrow[d] & b' \in B' \arrow[r] & C' & \\
& {\color{blue} a' + \mathrm{im}(\alpha)} \in \mathrm{coker}(\alpha) & & &
\end{tikzcd}$$

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

Connecting Homomorphism in the Snake Lemma

$$\begin{tikzcd}
 & & & c \in \ker(\gamma) \arrow[d] \arrow[dddll, rounded corners] & \\
 & A \arrow[r] \arrow[d, "\alpha"'] & b \in B \arrow[r, "\epsilon"] \arrow[d, "\beta"'] & c \in C \arrow[r] \arrow[d, "\gamma"'] & 0 \\
0 \arrow[r] & a' \in A' \arrow[r, "\mu"'] \arrow[d] & b' \in B' \arrow[r] & C' & \\
 & a' + \mathrm{im}(\alpha) \in \mathrm{coker}(\alpha) & & &
\end{tikzcd}$$

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

Connecting Homomorphism in the Snake Lemma

$$
\begin{tikzcd}
 & & & c \in \ker(\gamma) \arrow[d] \arrow[dddll, rounded corners] & \\
 & A \arrow[r] \arrow[d, "\alpha"'] & b \in B \arrow[r, "\epsilon"] \arrow[d, "\beta"'] & c \in C \arrow[r] \arrow[d, "\gamma"'] & 0 \\
0 \arrow[r] & a' \in A' \arrow[r, "\mu"'] \arrow[d] & b' \in B' \arrow[r] & C' & \\
 & a' + \mathrm{im}(\alpha) \in \mathrm{coker}(\alpha) & & &
\end{tikzcd}
$$

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

Connecting Homomorphism in the Snake Lemma

$$
\begin{tikzcd}
 & & & c \in \ker(\gamma) \arrow[d] \arrow[dddll, rounded corners] & \\
 & A \arrow[r] \arrow[d, "\alpha"'] & {\color{red}b} \in B \arrow[r, "\epsilon"] \arrow[d, "\beta"'] & c \in C \arrow[r] \arrow[d, "\gamma"'] & 0 \\
0 \arrow[r] & a' \in A' \arrow[r, "\mu"'] \arrow[d] & b' \in B' \arrow[r] & C' & \\
 & a' + \operatorname{im}(\alpha) \in \operatorname{coker}(\alpha) & & &
\end{tikzcd}
$$

Idea: Use **relations** instead of maps. $c \mapsto \epsilon^{-1}(\{b\})$

Relations

Let A, B be abelian groups.

Relations

Let A, B be abelian groups.

Definition

A subgroup $f \subseteq A \oplus B$ is called a **relation from A to B.**

Relations

Let A, B be abelian groups.

Definition

A subgroup $f \subseteq A \oplus B$ is called a **relation from** A **to** B.

Example

Let $\epsilon : B \to C$ be a homomorphism of abelian groups.

Relations

Let A, B be abelian groups.

Definition

A subgroup $f \subseteq A \oplus B$ is called a **relation from A to B**.

Example

Let $\epsilon : B \to C$ be a homomorphism of abelian groups.

$$\Gamma(\epsilon) := \{(b, c) \in B \oplus C \mid \epsilon(b) = c\}$$

is a relation from B to C

Relations

Let A, B be abelian groups.

Definition

A subgroup $f \subseteq A \oplus B$ is called a **relation from A to B**.

Example

Let $\epsilon : B \to C$ be a homomorphism of abelian groups.

$$\Gamma(\epsilon) := \{(b, c) \in B \oplus C \mid \epsilon(b) = c\}$$

is a relation from B to C, called **graph of** ϵ

Relations

Let A, B be abelian groups.

Definition

A subgroup $f \subseteq A \oplus B$ is called a **relation from A to B**.

Example

Let $\epsilon : B \to C$ be a homomorphism of abelian groups.

$$\Gamma(\epsilon) := \{(b, c) \in B \oplus C \mid \epsilon(b) = c\}$$

is a relation from B to C, called **graph of** ϵ, and

$$\epsilon^{-1} := \{(c, b) \in C \oplus B \mid \epsilon(b) = c\}$$

is a relation from C to B

Relations

Let A, B be abelian groups.

Definition

A subgroup $f \subseteq A \oplus B$ is called a **relation from A to B**.

Example

Let $\epsilon : B \to C$ be a homomorphism of abelian groups.

$$\Gamma(\epsilon) := \{(b, c) \in B \oplus C \mid \epsilon(b) = c\}$$

is a relation from B to C, called **graph of** ϵ, and

$$\epsilon^{-1} := \{(c, b) \in C \oplus B \mid \epsilon(b) = c\}$$

is a relation from C to B, called **pseudo-inverse**.

Relations

Composition of Relations

Relations

Composition of Relations

Given $f \subseteq A \oplus B$ and $g \subseteq B \oplus C$, define

Relations

Composition of Relations

Given $f \subseteq A \oplus B$ and $g \subseteq B \oplus C$, define

$$g \circ f := \{(a, c) \in A \oplus C \mid \exists b \in B : (a, b) \in f, (b, c) \in g\}$$

Relations

Composition of Relations

Given $f \subseteq A \oplus B$ and $g \subseteq B \oplus C$, define

$$g \circ f := \{(a, c) \in A \oplus C \mid \exists b \in B : (a, b) \in f, (b, c) \in g\}$$

If f and g correspond to maps, this describes their usual composition.

Snake Lemma Revisited

$$\begin{tikzcd}
 & & & \ker(\gamma) \arrow[d, "\iota"] & \\
 & A \arrow[r] \arrow[d, "\alpha"'] & B \arrow[r, "\epsilon"] \arrow[d, "\beta"'] & C \arrow[r] \arrow[d, "\gamma"'] & 0 \\
0 \arrow[r] & A' \arrow[r, "\mu"'] \arrow[d, "\pi"'] & B' \arrow[r] & C' & \\
 & \operatorname{coker}(\alpha) & & &
\end{tikzcd}$$

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

Snake Lemma Revisited

$$\begin{tikzcd}
 & & & \ker(\gamma) \arrow[d, "\iota"] & \\
 & A \arrow[r] \arrow[d, "\alpha"'] & B \arrow[r, "\epsilon"] \arrow[d, "\beta"'] & C \arrow[r] \arrow[d, "\gamma"'] & 0 \\
0 \arrow[r] & A' \arrow[r, "\mu"'] \arrow[d, "\pi"'] & B' \arrow[r] & C' & \\
 & \operatorname{coker}(\alpha) & & &
\end{tikzcd}$$

ι

Snake Lemma Revisited

$$
\begin{tikzcd}
 & & & \ker(\gamma) \arrow[d, "\iota"] & \\
 & A \arrow[r] \arrow[d, "\alpha"'] & B \arrow[r, "\epsilon"] \arrow[d, "\beta"'] & C \arrow[r] \arrow[d, "\gamma"'] \arrow[l, bend right, "\epsilon^{-1}"'] & 0 \\
0 \arrow[r] & A' \arrow[r, "\mu"'] \arrow[d, "\pi"'] & B' \arrow[r] & C' & \\
 & \mathrm{coker}(\alpha) & & &
\end{tikzcd}
$$

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

Snake Lemma Revisited

$$
\begin{tikzcd}
& & & \mathrm{ker}(\gamma) \arrow[d, "\iota"] & \\
& A \arrow[r] \arrow[d, "\alpha"'] & B \arrow[r, "\epsilon"] \arrow[d, "\beta"'] & C \arrow[r] \arrow[d, "\gamma"'] \arrow[l, bend right, "\epsilon^{-1}"'] & 0 \\
0 \arrow[r] & A' \arrow[r, "\mu"'] \arrow[d, "\pi"'] & B' \arrow[r] & C' & \\
& \mathrm{coker}(\alpha) & & &
\end{tikzcd}
$$

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

Snake Lemma Revisited

$$\begin{tikzcd}
 & & & \ker(\gamma) \arrow[d, "\iota"] & \\
 & A \arrow[r] \arrow[d, "\alpha"'] & B \arrow[r, "\epsilon"] \arrow[d, "\beta"'] & C \arrow[r] \arrow[d, "\gamma"'] \arrow[l, bend right, "\epsilon^{-1}"'] & 0 \\
0 \arrow[r] & A' \arrow[r, "\mu"'] \arrow[d, "\pi"'] & B' \arrow[r] \arrow[l, bend left, "\mu^{-1}"] & C' & \\
 & \operatorname{coker}(\alpha) & & &
\end{tikzcd}$$

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

Snake Lemma Revisited

$$
\begin{tikzcd}
 & & & \ker(\gamma) \arrow[d, blue, "\iota"] & \\
 & A \arrow[r] \arrow[d, "\alpha"'] & B \arrow[r, "\epsilon"] \arrow[d, blue, "\beta"'] & C \arrow[r] \arrow[d, "\gamma"'] \arrow[l, red, bend right, "\epsilon^{-1}"'] & 0 \\
0 \arrow[r] & A' \arrow[r, "\mu"'] \arrow[d, blue, "\pi"'] & B' \arrow[r] \arrow[l, red, bend left, "\mu^{-1}"] & C' & \\
 & \operatorname{coker}(\alpha) & & &
\end{tikzcd}
$$

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

Snake Lemma Revisited

$$\begin{tikzcd}
 & & & \ker(\gamma) \arrow[d, "\iota"] \arrow[dddll, rounded corners] & \\
 & A \arrow[r] \arrow[d, "\alpha"'] & B \arrow[r, "\epsilon"] \arrow[d, "\beta"'] & C \arrow[r] \arrow[d, "\gamma"'] \arrow[l, bend right, "\epsilon^{-1}"'] & 0 \\
0 \arrow[r] & A' \arrow[r, "\mu"'] \arrow[d, "\pi"'] & B' \arrow[r] \arrow[l, bend left, "\mu^{-1}"] & C' & \\
 & \mathrm{coker}(\alpha) & & &
\end{tikzcd}$$

∂ is a composition of relations!

From Relations to Generalized Morphisms

- **Wanted**: A categorical framework for relations.

From Relations to Generalized Morphisms

- **Wanted**: A categorical framework for relations.
- **Solution**: Generalized Morphisms.

From Relations to Generalized Morphisms

Let A, B be objects in an abelian category $\mathbf{A}$.

From Relations to Generalized Morphisms

Let A, B be objects in an abelian category $\mathbf{A}$.

Relation

$$\begin{tikzcd} A \oplus B \\ D \arrow[u, hook] \end{tikzcd}$$

From Relations to Generalized Morphisms

Let A, B be objects in an abelian category $\mathbf{A}$.

Relation

$$\begin{tikzcd} A \oplus B \\ D \arrow[u, hook, "\begin{pmatrix} \alpha & \beta \end{pmatrix}"] \end{tikzcd}$$

From Relations to Generalized Morphisms

Let A, B be objects in an abelian category **A**.

Relation

$$\begin{tikzcd} A & & B \\ & D \arrow[ul, "\alpha"] \arrow[ur, "\beta"'] & \end{tikzcd}$$

From Relations to Generalized Morphisms

Let A, B be objects in an abelian category **A**.

Relation $\rightsquigarrow$ Generalized Morphism

$$
\begin{tikzcd}
A \arrow[rr, dashed] & & B \\
& D \arrow[ul, "\alpha"] \arrow[ur, "\beta"'] &
\end{tikzcd}
$$

Composition of Generalized Morphisms

Composition

$$
\begin{tikzcd}
A \arrow[rr, dashed] & & B \\
& D \arrow[ul] \arrow[ur] &
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
B \arrow[rr, dashed] & & C \\
& E \arrow[ul] \arrow[ur] &
\end{tikzcd}
$$

Composition of Generalized Morphisms

Composition

$$\begin{tikzcd}
A & & B & & C \\
& D \arrow[ul] \arrow[ur] & & E \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

Composition of Generalized Morphisms

Composition

$$
\begin{tikzcd}
A & & B & & C \\
& D \arrow[ul] \arrow[ur] & & E \arrow[ul] \arrow[ur] & \\
& & D \times_B E \arrow[ul] \arrow[ur] & &
\end{tikzcd}
$$

Composition of Generalized Morphisms

Composition

$$\begin{tikzcd}
A & & B & & C \\
& D \arrow[ul] \arrow[ur] & & E \arrow[ul] \arrow[ur] & \\
& & D \times_B E \arrow[ul] \arrow[ur] & &
\end{tikzcd}$$

Composition of Generalized Morphisms

Composition

$$\begin{tikzcd}
A \arrow[rr, dashed, bend left] & & B & & C \\
& D \arrow[ul] \arrow[ur] & & E \arrow[ul] \arrow[ur] & \\
& & D \times_B E \arrow[ul] \arrow[ur] & &
\end{tikzcd}$$

Composition of Generalized Morphisms

Composition

$$
\begin{tikzcd}
A \arrow[rrrr, dashed, bend left] & & B & & C \\
& D \arrow[ul] \arrow[ur] & & E \arrow[ul] \arrow[ur] & \\
& & D \times_B E \arrow[ul] \arrow[ur] & &
\end{tikzcd}
$$

Pullbacks $\rightsquigarrow$ Composition of generalized morphisms

Pseudo-Inverses

Pseudo-Inverses

$$\begin{tikzcd} A \arrow[rr, dashed] & & B \\ & D \arrow[ul] \arrow[ur] & \end{tikzcd}$$

Pseudo-Inverses

Pseudo-Inverses

$$\begin{tikzcd} A \arrow[rr, dashed] & & B \\ & D \arrow[ul] \arrow[ur] & \end{tikzcd}$$

Pseudo-Inverses

Pseudo-Inverses

$$\begin{tikzcd}
A \arrow[rr, dashed] & & B \\
& D \arrow[ul, red] \arrow[ur, blue] \arrow[u, phantom, "\leftrightarrow"] &
\end{tikzcd}$$

Pseudo-Inverses

Pseudo-Inverses

$$\begin{tikzcd} A \arrow[rr, dashed] & & B \\ & D \arrow[ul, red] \arrow[ur, blue] \arrow[u, phantom, "\leftrightarrow"] & \end{tikzcd} \qquad \rightsquigarrow$$

Pseudo-Inverses

Pseudo-Inverses

$$
\begin{tikzcd}
A \arrow[rr, dashed] & & B \\
& D \arrow[ul, red] \arrow[ur, blue] \arrow[u, phantom, "\leftrightarrow"] &
\end{tikzcd}
\quad \rightsquigarrow \quad
\begin{tikzcd}
B \arrow[rr, dashed] & & A \\
& D \arrow[ul, blue] \arrow[ur, red] &
\end{tikzcd}
$$

Generalized Morphisms in CAP

Realization in CAP

- CAP provides generalized morphisms as a data structure.

Generalized Morphisms in CAP

Realization in CAP

- CAP provides generalized morphisms as a data structure.
- 3 different data structures:

Generalized Morphisms in CAP

Realization in CAP

- CAP provides generalized morphisms as a data structure.
- 3 different data structures:
  1. Spans: $A \leftarrow D \rightarrow B$

Generalized Morphisms in CAP

Realization in CAP

- CAP provides generalized morphisms as a data structure.
- 3 different data structures:
  1. Spans: $A \leftarrow D \rightarrow B$
  2. Cospans: $A \rightarrow D \leftarrow B$

Generalized Morphisms in CAP

Realization in CAP

- CAP provides generalized morphisms as a data structure.
- 3 different data structures:
  1. Spans: $A \leftarrow D \rightarrow B$
  2. Cospans: $A \rightarrow D \leftarrow B$
  3. 3-arrow: $A \leftarrow D \rightarrow E \leftarrow B$

Generalized Morphisms in CAP

Realization in CAP

- CAP provides generalized morphisms as a data structure.
- 3 different data structures:
  1. Spans: $A \leftarrow D \rightarrow B$
  2. Cospans: $A \rightarrow D \leftarrow B$
  3. 3-arrow: $A \leftarrow D \rightarrow E \leftarrow B$
- They are useful for:

Generalized Morphisms in CAP

Realization in CAP

- CAP provides generalized morphisms as a data structure.
- 3 different data structures:
  1. Spans: $A \leftarrow D \rightarrow B$
  2. Cospans: $A \rightarrow D \leftarrow B$
  3. 3-arrow: $A \leftarrow D \rightarrow E \leftarrow B$
- They are useful for:
  1. Diagram chases

Generalized Morphisms in CAP

Realization in CAP

- CAP provides generalized morphisms as a data structure.
- 3 different data structures:
  1. Spans: $A \leftarrow D \rightarrow B$
  2. Cospans: $A \rightarrow D \leftarrow B$
  3. 3-arrow: $A \leftarrow D \rightarrow E \leftarrow B$
- They are useful for:
  1. Diagram chases
  2. Spectral sequences

Generalized Morphisms in CAP

Realization in CAP

- CAP provides generalized morphisms as a data structure.
- 3 different data structures:
    1. Spans: $A \leftarrow D \rightarrow B$
    2. Cospans: $A \rightarrow D \leftarrow B$
    3. 3-arrow: $A \leftarrow D \rightarrow E \leftarrow B$
- They are useful for:
    1. Diagram chases
    2. Spectral sequences
    3. Localization of categories

Generalized Morphisms in CAP

Realization in CAP

- CAP provides generalized morphisms as a data structure.
- 3 different data structures:
  1. Spans: $A \leftarrow D \rightarrow B$
  2. Cospans: $A \rightarrow D \leftarrow B$
  3. 3-arrow: $A \leftarrow D \rightarrow E \leftarrow B$
- They are useful for:
  1. Diagram chases
  2. Spectral sequences
  3. Localization of categories (Serre quotients)

Download CAP

Download CAP

http://homalg-project.github.io/CAP_project/