First steps in CAP

Sebastian Gutsche, Sebastian Posur

Siegen University

August 19, 2018

Abstraction of language

Computing the intersection of two subobjects

Abstraction of language

Computing the intersection of two subobjects

Vector spaces

$\langle v_1, v_2 \rangle, \langle w_1, w_2 \rangle \leq V$:

Abstraction of language

Computing the intersection of two subobjects

Vector spaces

$\langle v_1, v_2 \rangle, \langle w_1, w_2 \rangle \leq V$:
Solution of

$$
\begin{aligned}
&x_1 v_1 + x_2 v_2 \\
=&y_1 w_1 + y_2 w_2
\end{aligned}
$$

Abstraction of language

Computing the intersection of two subobjects

Vector spaces

$\langle v_1, v_2 \rangle, \langle w_1, w_2 \rangle \leq V$:
Solution of

$$\begin{aligned} & x_1 v_1 + x_2 v_2 \\ = & y_1 w_1 + y_2 w_2 \end{aligned}$$

Ideals of $\mathbb{Z}$

Abstraction of language

Computing the intersection of two subobjects

Vector spaces

$\langle v_1, v_2\rangle, \langle w_1, w_2\rangle \leq V$:
Solution of

$$\begin{aligned} & x_1 v_1 + x_2 v_2 \\ = & y_1 w_1 + y_2 w_2 \end{aligned}$$

Ideals of $\mathbb{Z}$

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

Abstraction of language

Computing the intersection of two subobjects

Vector spaces

$\langle v_1, v_2 \rangle, \langle w_1, w_2 \rangle \leq V$:
Solution of

$$\begin{aligned} &x_1 v_1 + x_2 v_2 \\ =&y_1 w_1 + y_2 w_2 \end{aligned}$$

Ideals of $\mathbb{Z}$

$\langle x \rangle, \langle y \rangle \leq \mathbb{Z}$:
Euclidean algorithm:

$$\langle \operatorname{lcm}(x, y) \rangle$$

Abstraction of language

Computing the intersection of two subobjects

Vector spaces

$\langle v_1, v_2 \rangle, \langle w_1, w_2 \rangle \leq V$:
Solution of

$$x_1 v_1 + x_2 v_2 = y_1 w_1 + y_2 w_2$$

Ideals of $\mathbb{Z}$

$\langle x \rangle, \langle y \rangle \leq \mathbb{Z}$:
Euclidean algorithm:

$$\langle \mathrm{lcm}(x, y) \rangle$$

Generic algorithm for both cases?

Abstraction of language

Computing the intersection of two subobjects

Vector spaces

$\langle v_1, v_2 \rangle, \langle w_1, w_2 \rangle \leq V$:
Solution of

$$\begin{aligned} & x_1 v_1 + x_2 v_2 \\ = & y_1 w_1 + y_2 w_2 \end{aligned}$$

Ideals of $\mathbb{Z}$

$\langle x \rangle, \langle y \rangle \leq \mathbb{Z}$:
Euclidean algorithm:

$$\langle \operatorname{lcm}(x, y) \rangle$$

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

Computing the intersection

Let $M_1 \subseteq N$ and $M_2 \subseteq N$ subobjects.

Computing the intersection

Let $M_1 \hookrightarrow N$ and $M_2 \hookrightarrow N$ subobjects.

Computing the intersection

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

Computing the intersection

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

$$
\begin{tikzcd}
M_1 \arrow[dr, hook, "\iota_1"] & \\
& N \\
M_2 \arrow[ur, hook, "\iota_2"'] &
\end{tikzcd}
$$

Computing the intersection

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

$$\begin{tikzcd}
 & M_1 \arrow[dr, hook, "\iota_1"] & \\
M_1 \oplus M_2 & & N \\
 & M_2 \arrow[ur, hook, "\iota_2"'] &
\end{tikzcd}$$

Computing the intersection

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

$$
\begin{tikzcd}
 & M_1 \arrow[dr, hook, "\iota_1"] & \\
M_1 \oplus M_2 \arrow[ur, two heads, "\pi_1"] & & N \\
 & M_2 \arrow[ur, hook, "\iota_2"'] &
\end{tikzcd}
$$

Computing the intersection

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

$$
\begin{tikzcd}
 & M_1 \arrow[dr, hook, "\iota_1"] & \\
M_1 \oplus M_2 \arrow[ur, two heads, "\pi_1"] \arrow[dr, two heads, "\pi_2"'] & & N \\
 & M_2 \arrow[ur, hook, "\iota_2"'] &
\end{tikzcd}
$$

Computing the intersection

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

$$\begin{tikzcd}
 & M_1 \arrow[dr, hook, "\iota_1"] & \\
M_1 \oplus M_2 \arrow[ur, two heads, "\pi_1"] \arrow[dr, two heads, "\pi_2"'] & & N \\
 & M_2 \arrow[ur, hook, "\iota_2"'] &
\end{tikzcd}$$

- $\pi_i := \mathrm{ProjectionInFactorOfDirectSum}\left((M_1, M_2), i\right)$, $i = 1, 2$

Computing the intersection

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

$$\begin{tikzcd}
 & M_1 \arrow[dr, hook, "\iota_1"] & \\
M_1 \oplus M_2 \arrow[ur, two heads, "\pi_1"] \arrow[rr, "\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2"] \arrow[dr, two heads, "\pi_2"'] & & N \\
 & M_2 \arrow[ur, hook, "\iota_2"'] &
\end{tikzcd}$$

- $\pi_i := \mathrm{ProjectionInFactorOfDirectSum}\left((M_1, M_2), i\right)$, $i = 1, 2$

Computing the intersection

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

$$
\begin{tikzcd}
 & M_1 \arrow[dr, hook, "\iota_1"] & \\
M_1 \oplus M_2 \arrow[ur, two heads, "\pi_1"] \arrow[rr, "\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2"] \arrow[dr, two heads, "\pi_2"'] & & N \\
 & M_2 \arrow[ur, hook, "\iota_2"'] &
\end{tikzcd}
$$

- $\pi_i := \mathrm{ProjectionInFactorOfDirectSum}\left((M_1, M_2), i\right)$, $i = 1, 2$
- $\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$

Computing the intersection

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

$$
\begin{tikzcd}
 & & M_1 \arrow[dr, hook, "\iota_1"] & \\
M_1 \cap M_2 \arrow[r, hook, "\kappa"] & M_1 \oplus M_2 \arrow[ur, two heads, "\pi_1"] \arrow[dr, two heads, "\pi_2"'] \arrow[rr, "\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2"] & & N \\
 & & M_2 \arrow[ur, hook, "\iota_2"'] &
\end{tikzcd}
$$

- $\pi_i := \mathrm{ProjectionInFactorOfDirectSum}\left((M_1, M_2), i\right)$, $i = 1, 2$
- $\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$

Computing the intersection

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

$$
\begin{tikzcd}
 & & M_1 \arrow[dr, hook, "\iota_1"] & \\
M_1 \cap M_2 \arrow[r, hook, "\kappa"] & M_1 \oplus M_2 \arrow[ur, two heads, "\pi_1"] \arrow[rr, "\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2"] \arrow[dr, two heads, "\pi_2"'] & & N \\
 & & M_2 \arrow[ur, hook, "\iota_2"'] &
\end{tikzcd}
$$

- $\pi_i := \mathrm{ProjectionInFactorOfDirectSum}\left((M_1, M_2), i\right)$, $i = 1, 2$
- $\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$
- $\kappa := \mathrm{KernelEmbedding}\left(\varphi\right)$

Computing the intersection

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

$$\begin{tikzcd}
 & & M_1 \arrow[dr, hook, "\iota_1"] & \\
M_1 \cap M_2 \arrow[r, hook, "\kappa"] & M_1 \oplus M_2 \arrow[ur, two heads, "\pi_1"] \arrow[dr, two heads, "\pi_2"'] \arrow[rr, "\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2"] & & N \\
 & & M_2 \arrow[ur, hook, "\iota_2"'] &
\end{tikzcd}$$

- $\pi_i := \mathrm{ProjectionInFactorOfDirectSum}\left((M_1, M_2), i\right)$, $i = 1, 2$
- $\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$
- $\kappa := \mathrm{KernelEmbedding}\left(\varphi\right)$

Computing the intersection

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

$$
\begin{tikzcd}
 & & M_1 \arrow[dr, hook, "\iota_1"] & \\
M_1 \cap M_2 \arrow[r, hook, "\kappa"] & M_1 \oplus M_2 \arrow[ur, two heads, "\pi_1"] \arrow[dr, two heads, "\pi_2"'] \arrow[rr, "\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2"] & & N \\
 & & M_2 \arrow[ur, hook, "\iota_2"'] &
\end{tikzcd}
$$

- $\pi_i := \mathrm{ProjectionInFactorOfDirectSum}\left((M_1, M_2), i\right)$, $i = 1, 2$
- $\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$
- $\kappa := \mathrm{KernelEmbedding}\left(\varphi\right)$
- $\gamma := \iota_1 \circ \pi_1 \circ \kappa$

Translation to CAP

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

$\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$

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

$\gamma := \iota_1 \circ \pi_1 \circ \kappa$

Translation to CAP

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

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

$\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$

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

$\gamma := \iota_1 \circ \pi_1 \circ \kappa$

Translation to CAP

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

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

$\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$

```
lambda := PostCompose( iota1, pi1 );
phi := lambda - PostCompose( iota2, pi2 );
```

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

$\gamma := \iota_1 \circ \pi_1 \circ \kappa$

Translation to CAP

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

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

$\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$

```
lambda := PostCompose( iota1, pi1 );
phi := lambda - PostCompose( iota2, pi2 );
```

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

```
kappa := KernelEmbedding( phi );
```

$\gamma := \iota_1 \circ \pi_1 \circ \kappa$

Translation to CAP

$\pi_i := \mathrm{ProjectionInFactorOfDirectSum}\left((M_1, M_2), i\right), i = 1, 2$

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

$\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$

```
lambda := PostCompose( iota1, pi1 );
phi := lambda - PostCompose( iota2, pi2 );
```

$\kappa := \mathrm{KernelEmbedding}(\varphi)$

```
kappa := KernelEmbedding( phi );
```

$\gamma := \iota_1 \circ \pi_1 \circ \kappa$

```
gamma := PostCompose( lambda, kappa );
```

Translation to CAP

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

lambda := PostCompose( iota1, pi1 );
phi := lambda - PostCompose( iota2, pi2 );

kappa := KernelEmbedding( phi );

gamma := PostCompose( lambda, kappa );
```

Translation to CAP

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

lambda := PostCompose( iota1, pi1 );
phi := lambda - PostCompose( iota2, pi2 );

kappa := KernelEmbedding( phi );

gamma := PostCompose( lambda, kappa );
```

Translation to CAP

```
IntersectionOfSubobjects := function( iota1, iota2 )
```

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

  lambda := PostCompose( iota1, pi1 );
  phi := lambda - PostCompose( iota2, pi2 );

  kappa := KernelEmbedding( phi );

  gamma := PostCompose( lambda, kappa );
```

Translation to CAP

```
IntersectionOfSubobjects := function( iota1, iota2 )

  M1 := Source( iota1 );
  M2 := Source( iota2 );
  pi1 := ProjectionInFactorOfDirectSum( [ M1, M2 ], 1 );
  pi2 := ProjectionInFactorOfDirectSum( [ M1, M2 ], 2 );
  lambda := PostCompose( iota1, pi1 );
  phi := lambda - PostCompose( iota2, pi2 );
  kappa := KernelEmbedding( phi );
  gamma := PostCompose( lambda, kappa );
```

Translation to CAP

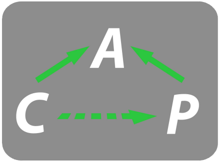
```
IntersectionOfSubobjects := function( iota1, iota2 )

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

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

  lambda := PostCompose( iota1, pi1 );
  phi := lambda - PostCompose( iota2, pi2 );

  kappa := KernelEmbedding( phi );

  gamma := PostCompose( lambda, kappa );

  return gamma;
end;
```

Translation to CAP

```
IntersectionOfSubobjects := function( iota1, iota2 )
  local M1, M2, pi1, pi2, lambda, phi, kappa, gamma;
  M1 := Source( iota1 );
  M2 := Source( iota2 );
  pi1 := ProjectionInFactorOfDirectSum( [ M1, M2 ], 1 );
  pi2 := ProjectionInFactorOfDirectSum( [ M1, M2 ], 2 );
  lambda := PostCompose( iota1, pi1 );
  phi := lambda - PostCompose( iota2, pi2 );
  kappa := KernelEmbedding( phi );
  gamma := PostCompose( lambda, kappa );
  return gamma;
end;
```

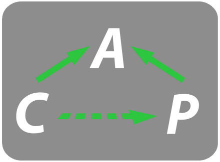
What is CAP?

What is CAP?



CAP means **Categories, Algorithms, Programming**

What is CAP?



CAP means **Categories, Algorithms, Programming** and is a software project implemented in GAP.

What is CAP?

CAP means **Categories, Algorithms, 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, 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, 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.

We call this concept **categorical programming**.

Tasks for today

1. Implementation of a category
2. Write a function for homology
3. Homework

Tasks for today

1. Implementation of a category
2. Write a function for homology
3. Homework

Computable categories

We implement a category by providing

Computable categories

We implement a category by providing

- *data structures* for objects and morphisms,

Computable categories

We implement a category by providing

- *data structures* for objects and morphisms,
- *algorithms* for basic categorical operations.

Computable categories

We implement a category by providing

- *data structures* for objects and morphisms,
- *algorithms* for basic categorical operations.

Example implementation: the category of groups

$\mathbb{Q}$-vector spaces

$\mathbb{Q}$-vector spaces

$\mathbb{Q}$-vector spaces (classical model)

- $\mathrm{Obj} :=$ finite dimensional $\mathbb{Q}$-vector spaces

$\mathbb{Q}$-vector spaces

$\mathbb{Q}$-vector spaces (classical model)

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

$\mathbb{Q}$-vector spaces

$\mathbb{Q}$-vector spaces (classical model)

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

$$\simeq$$

$\mathbb{Q}$-vector spaces

$\mathbb{Q}$-vector spaces (classical model)

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

$$\simeq$$

Matrices (computerfriendly model)

$\mathbb{Q}$-vector spaces

$\mathbb{Q}$-vector spaces (classical model)

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

$$\simeq$$

Matrices (computerfriendly model)

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

$\mathbb{Q}$-vector spaces

$\mathbb{Q}$-vector spaces (classical model)

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

$$\simeq$$

Matrices (computerfriendly model)

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

$\mathbb{Q}$-vector spaces

$\mathbb{Q}$-vector spaces (classical model)

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

$$\simeq$$

Matrices (computerfriendly model)

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

$\mathbb{Q}$-vector spaces

$\mathbb{Q}$-vector spaces (classical model)

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

$$\simeq$$

Matrices (computerfriendly model)

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

$\mathbb{Q}$-vector spaces

Computing in the computerfriendly model

$\mathbb{Q}$-vector spaces

Computing in the computerfriendly model

$1 \qquad 2 \qquad 1$

$\mathbb{Q}$-vector spaces

Computing in the computerfriendly model

1 2 1

$\mathbb{Q}$-vector spaces

Computing in the computerfriendly model

$$1 \xrightarrow[\begin{pmatrix} 1 & 2 \end{pmatrix}]{} 2 \xrightarrow[\begin{pmatrix} 3 \\ 4 \end{pmatrix}]{} 1$$

$\mathbb{Q}$-vector spaces

Computing in the computerfriendly model

$$1 \xrightarrow[\begin{pmatrix} 1 & 2 \end{pmatrix}]{} 2 \xrightarrow[\begin{pmatrix} 3 \\ 4 \end{pmatrix}]{} 1$$

$\mathbb{Q}$-vector spaces

Computing in the computerfriendly model

$$\begin{pmatrix} 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 4 \end{pmatrix} = (11)$$

$$\begin{tikzcd} 1 \arrow[r, "\begin{pmatrix} 1 & 2 \end{pmatrix}"'] \arrow[rr, bend left] & 2 \arrow[r, "\begin{pmatrix} 3 \\ 4 \end{pmatrix}"'] & 1 \end{tikzcd}$$

$\mathbb{Q}$-vector spaces

Computing in the computerfriendly model

$$\begin{pmatrix} 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 4 \end{pmatrix} = (11)$$

$$\begin{tikzcd}
1 \arrow[r, "\begin{pmatrix} 1 & 2 \end{pmatrix}"'] \arrow[rr, bend left] & 2 \arrow[r, "\begin{pmatrix} 3 \\ 4 \end{pmatrix}"'] & 1
\end{tikzcd}$$

$\mathbb{Q}$-vector spaces

Computing in the computerfriendly model

$$\begin{pmatrix}1 & 2\end{pmatrix}\cdot\begin{pmatrix}3\\4\end{pmatrix} = (11)$$

$$\begin{tikzcd} 1 \arrow[r, "\begin{pmatrix}1 & 2\end{pmatrix}"'] \arrow[rr, bend left] \arrow[loop below, "(1)"] & 2 \arrow[r, "\begin{pmatrix}3\\4\end{pmatrix}"'] \arrow[loop below, "\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}"] & 1 \arrow[loop below, "(1)"] \end{tikzcd}$$

$\mathbb{Q}$-vector spaces

Computing in the computerfriendly model

$$\begin{pmatrix} 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 4 \end{pmatrix} = (11)$$

$$\begin{tikzcd} 1 \arrow[loop below, "(1)"] \arrow[r, "\begin{pmatrix} 1 & 2 \end{pmatrix}"'] \arrow[rr, bend left] & 2 \arrow[loop below, "\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}"] \arrow[r, "\begin{pmatrix} 3 \\ 4 \end{pmatrix}"'] & 1 \arrow[loop below, "(1)"] \end{tikzcd}$$

$\mathbb{Q}$-vector spaces

Computing in the computerfriendly model

$$\begin{pmatrix} 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 4 \end{pmatrix} = (11)$$

$$\begin{tikzcd}
1 \arrow[r, "\begin{pmatrix} 1 & 2 \end{pmatrix}"'] \arrow[rr, bend left] \arrow[loop below, "(1)"] & 2 \arrow[r, "\begin{pmatrix} 3 \\ 4 \end{pmatrix}"'] \arrow[loop below, "\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}"] & 1 \arrow[loop below, "(1)"]
\end{tikzcd}$$

Download the task file

https://homalg-project.github.io/capdays-2018/materials/session01/HandsOnExercise.gi

Implementation of the kernel

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

Implementation of the kernel

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

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

Implementation of the kernel

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

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

Implementation of the kernel

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

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

$$\ker \varphi$$

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

Implementation of the kernel

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

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

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

Implementation of the kernel

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

... one needs an object $\ker \varphi$,
its embedding $\kappa = \mathrm{KernelEmbedding}(\varphi)$,
and for every test morphism τ

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

Implementation of the kernel

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

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

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

Implementation of the kernel

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

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

$$
\begin{tikzcd}
\ker \varphi \arrow[dr, hook, "\kappa"] \arrow[drr, bend left, "0"] & & \\
& A \arrow[r, "\varphi"] & B \\
T \arrow[uu, dashed, "\lambda"] \arrow[ur, "\tau"'] \arrow[urr, bend right, "0"'] \arrow[u, phantom, "\circlearrowright"] & &
\end{tikzcd}
$$

Download the task file

`https://homalg-project.github.io/capdays-2018/materials/session01/HandsOnExercise.gi`

Useful commands for homalg matrices

- $\texttt{HomalgZeroMatrix}(\mathtt{m}, \mathtt{n}, \mathbb{Q}) = 0^{m \times n}$
- Arithmetics: $*, +, -$
- $\texttt{SyzygiesOfColumns}(\mathtt{A}) =$ column kernel of $\mathtt{A}$
- $\mathtt{A} * \texttt{LeftDivide}(\mathtt{A}, \mathtt{B}) = \mathtt{B}$
- $\texttt{NrColumns}(\mathtt{A}) =$ number of columns of $\mathtt{A}$

Computing the intersection

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

$$\begin{tikzcd}
 & & M_1 \arrow[dr, hook, "\iota_1"] & \\
M_1 \cap M_2 \arrow[r, hook, "\kappa"] & M_1 \oplus M_2 \arrow[ur, two heads, "\pi_1"] \arrow[dr, two heads, "\pi_2"'] \arrow[rr, "\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2"] & & N \\
 & & M_2 \arrow[ur, hook, "\iota_2"'] &
\end{tikzcd}$$

- $\pi_i := \mathrm{ProjectionInFactorOfDirectSum}\left((M_1, M_2), i\right)$, $i = 1, 2$
- $\varphi := \iota_1 \circ \pi_1 - \iota_2 \circ \pi_2$
- $\kappa := \mathrm{KernelEmbedding}(\varphi)$
- $\gamma := \iota_1 \circ \pi_1 \circ \kappa$

Tasks for today

1. Implementation of a category
2. Write a function for homology
3. Homework

Homology

$$A \xrightarrow{\alpha} B \xrightarrow{\beta} C$$

Homology

$$
\begin{tikzcd}
A \arrow[rr, "\alpha"] & & B \arrow[rr, "\beta"] & & C \\
& \mathrm{im}\,(\alpha) \arrow[ur, hook, "\iota"] & & &
\end{tikzcd}
$$

Homology

$$\begin{tikzcd} A \arrow[r, "\alpha"] & B \arrow[r, "\beta"] & C \\ \operatorname{im}(\alpha) \arrow[ur, hook, "\iota"] & & \end{tikzcd}$$

Homology

$$\begin{tikzcd}
A \arrow[rr, "\alpha"] & & B \arrow[rr, "\beta"] & & C \\
& \mathrm{im}(\alpha) \arrow[ur, hook, "\iota"] & & \color{red}{\ker(\beta)} \arrow[ul, hook', red, "\kappa"'] &
\end{tikzcd}$$

Homology

$$\begin{tikzcd}
A \arrow[rr, "\alpha"] & & B \arrow[rr, "\beta"] & & C \\
& \mathrm{im}(\alpha) \arrow[ur, hook, "\iota"] & & \ker(\beta) \arrow[ul, hook', "\kappa"'] &
\end{tikzcd}$$

Homology

$$\begin{tikzcd}
A \arrow[rr, "\alpha"] & & B \arrow[rr, "\beta"] & & C \\
& \mathrm{im}(\alpha) \arrow[ur, hook, "\iota"] \arrow[rr, hook, red, "\gamma"'] & & \ker(\beta) \arrow[ul, hook', "\kappa"'] &
\end{tikzcd}$$

Homology

$$
\begin{tikzcd}
A \arrow[rr, "\alpha"] & & B \arrow[rr, "\beta"] & & C \\
& \mathrm{im}\,(\alpha) \arrow[ur, hook, "\iota"] \arrow[rr, hook, "\gamma"'] & & \ker\,(\beta) \arrow[ul, hook', "\kappa"'] &
\end{tikzcd}
$$

Homology

$$\begin{tikzcd}
A \arrow[rr, "\alpha"] & & B \arrow[rr, "\beta"] & & C \\
& \mathrm{im}(\alpha) \arrow[ur, hook, "\iota"] \arrow[rr, hook, "\gamma"'] & & \ker(\beta) \arrow[ul, hook', "\kappa"'] \arrow[r, two heads, red] & \textcolor{red}{H}
\end{tikzcd}$$

Homology

$$
\begin{tikzcd}
A \arrow[rr, "\alpha"] & & B \arrow[rr, "\beta"] & & C \\
& \mathrm{im}\,(\alpha) \arrow[ur, hook, "\iota"] \arrow[rr, hook, "\gamma"'] & & \ker\,(\beta) \arrow[ul, hook', "\kappa"'] \arrow[r, two heads] & \mathrm{H}
\end{tikzcd}
$$

Download the task file

`https://homalg-project.github.io/capdays-2018/materials/session01/HandsOnExercise.gi`

Useful CAP commands

- $\mathtt{ImageEmbedding}(\mathtt{A} \xrightarrow{\alpha} \mathtt{B}) =$ $$\begin{tikzcd} \mathtt{A} \arrow[rr] \arrow[dr] & & \mathtt{B} \\ & \mathtt{im}(\alpha) \arrow[ur, hook] & \end{tikzcd}$$

- $\mathtt{KernelLift}(\mathtt{A} \xrightarrow{\alpha} \mathtt{B},\ \mathtt{T} \xrightarrow{\tau} \mathtt{A}) =$ $$\begin{tikzcd} \mathtt{ker}(\alpha) \arrow[r, hook] & \mathtt{A} \\ \mathtt{T} \arrow[u, dashed] \arrow[ur, "\tau"'] & \end{tikzcd}$$

- $\mathtt{CokernelObject}(\mathtt{A} \xrightarrow{\alpha} \mathtt{B}) = \mathtt{B} \twoheadrightarrow \mathtt{coker}(\alpha)$

Homology: solution

$$\begin{tikzcd} A \arrow[rr, "\alpha"] & & B \arrow[rr, "\beta"] & & C \\ & \mathrm{im}\,(\alpha) \arrow[ur, hook, "\iota"] \arrow[rr, hook, "\gamma"'] & & \ker\,(\beta) \arrow[ul, hook', "\kappa"'] \arrow[r, two heads] & \mathrm{H} \end{tikzcd}$$

Homology: solution

$$
\begin{tikzcd}
A \arrow[rr, "\alpha"] & & B \arrow[rr, "\beta"] & & C \\
& \operatorname{im}(\alpha) \arrow[ur, hook, "\iota"] \arrow[rr, hook, "\gamma"'] & & \ker(\beta) \arrow[ul, hook', "\kappa"'] \arrow[r, two heads] & \mathrm{H}
\end{tikzcd}
$$

```
HomologyObject := function( alpha, beta )
  local iota, gamma;

  iota := ImageEmbedding( alpha );
  gamma := KernelLift( beta, iota );
  return CokernelObject( gamma );

end;
```

Tasks for today

1. Implementation of a category
2. Write a function for homology
3. Homework

Snake lemma

Write a function for the connecting homomorphism.

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

Snake lemma

Write a function for the connecting homomorphism.

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

Snake lemma

Write a function for the connecting homomorphism.

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

What input is relevant for the construction?