CAP in QPA

Øystein Skartsæterhagen

Department of Mathematical Sciences
Norwegian University of Science and Technology

First CAP Days, 2018-08-28

(Parts of this presentation are adapted from *Introduction to QPA*,
my joint talk with Øyvind Solberg at the Third GAP Days in Trondheim, 2015)

What is QPA?

- QPA: “Quivers and Path Algebras”
- GAP package for computations with quotients of path algebras and their modules

Part 1

So what are these “quivers”, anyway?

Quivers

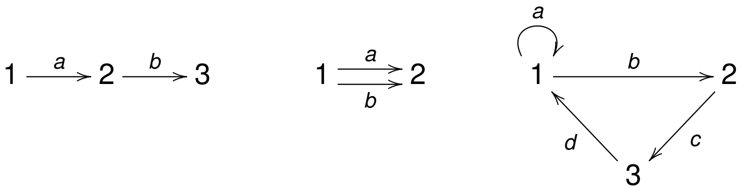
Quivers

$$\begin{tikzcd} 1 \arrow[r, "a"] & 2 \arrow[r, "b"] & 3 \end{tikzcd}$$

$$\begin{tikzcd} 1 \arrow[r, shift left, "a"] \arrow[r, shift right, "b"'] & 2 \end{tikzcd}$$

$$\begin{tikzcd} 1 \arrow[loop above, "a"] \arrow[rr, "b"] & & 2 \arrow[dl, "c"] \\ & 3 \arrow[ul, "d"] & \end{tikzcd}$$

Quivers



Quiver: oriented graph (loops and multiple edges allowed)

Paths

$$Q\colon\ 1 \xrightarrow{a} 2 \xrightarrow{b} 3$$

Paths

$$Q\colon\ 1 \xrightarrow{a} 2 \xrightarrow{b} 3$$

Paths in Q:

Paths

$$Q\colon\ 1 \xrightarrow{a} 2 \xrightarrow{b} 3$$

Paths in Q:

- Length 0: e_1, e_2, e_3 (vertices/trivial paths)

Paths

$$Q\colon\ 1 \xrightarrow{a} 2 \xrightarrow{b} 3$$

Paths in Q:

- Length 0: e_1, e_2, e_3 (vertices/trivial paths)
- Length 1: a, b (arrows)

Paths

$$Q\colon\ 1 \xrightarrow{a} 2 \xrightarrow{b} 3$$

Paths in Q:

- Length 0: e_1, e_2, e_3 (vertices/trivial paths)
- Length 1: a, b (arrows)
- Length 2: ba (concatenation of a and b)

Path algebras

$$\begin{tikzcd} Q\colon & 1 \arrow[r, "a"] & 2 \arrow[r, "b"] & 3 \end{tikzcd}$$

k a field

Path algebras

$$\left.\begin{array}{l} Q\colon\ 1 \xrightarrow{a} 2 \xrightarrow{b} 3 \\ k \text{ a field} \end{array}\right\} \rightsquigarrow \text{path algebra } kQ$$

Path algebras

$$\left.\begin{array}{l} Q\colon\ 1 \xrightarrow{a} 2 \xrightarrow{b} 3 \\ k \text{ a field} \end{array}\right\} \rightsquigarrow \text{path algebra } kQ$$

- Basis: $\{e_1, e_2, e_3, a, b, ba\}$

Path algebras

$$\left.\begin{array}{l} Q\colon\ 1 \xrightarrow{a} 2 \xrightarrow{b} 3 \\ k \text{ a field} \end{array}\right\} \rightsquigarrow \text{path algebra } kQ$$

- Basis: $\{e_1, e_2, e_3, a, b, ba\}$
- Multiplication:

$$\begin{aligned} e_1 \cdot e_1 &= e_1 \\ e_1 \cdot e_2 &= 0 \\ e_2 \cdot a &= a \\ e_1 \cdot a &= 0 \\ b \cdot a &= ba \end{aligned}$$

Path algebras

$$\left.\begin{array}{l} Q\colon\ 1 \xrightarrow{a} 2 \xrightarrow{b} 3 \\ k \text{ a field} \end{array}\right\} \rightsquigarrow \text{path algebra } kQ$$

- Basis: $\{e_1, e_2, e_3, a, b, ba\}$
- Multiplication:

$$e_1 \cdot e_1 = e_1$$

$$e_1 \cdot e_2 = 0$$

$$e_2 \cdot a = a$$

$$e_1 \cdot a = 0$$

$$b \cdot a = ba$$

$\cdot$	e_1	e_2	e_3	a	b	ba
e_1	e_1	0	0	0	0	0
e_2	0	e_2	0	a	0	0
e_3	0	0	e_3	0	b	ba
a	a	0	0	0	0	0
b	0	b	0	ba	0	0
ba	ba	0	0	0	0	0

Representations

Given

$$Q\colon\ 1 \xrightarrow{a} 2 \xrightarrow{b} 3$$

Want to make a representation R of Q.

Representations

Given

$$\begin{tikzcd} Q\colon & 1 \arrow[r, "a"] & 2 \arrow[r, "b"] & 3 \end{tikzcd}$$

Want to make a representation R of Q.

$$\begin{tikzcd} R\colon & \bullet \arrow[r] & \bullet \arrow[r] & \bullet \end{tikzcd}$$

Start with the quiver, and put

Representations

Given

$$\begin{tikzcd} Q\colon\ 1 \arrow[r, "a"] & 2 \arrow[r, "b"] & 3 \end{tikzcd}$$

Want to make a representation R of Q.

$$\begin{tikzcd} R\colon\ V_1 \arrow[r] & V_2 \arrow[r] & V_3 \end{tikzcd}$$

Start with the quiver, and put

— a vector space at each vertex,

Representations

Given

$$Q\colon\ 1 \xrightarrow{a} 2 \xrightarrow{b} 3$$

Want to make a representation R of Q.

$$R\colon\ V_1 \xrightarrow{f_a} V_2 \xrightarrow{f_b} V_3$$

Start with the quiver, and put

- — a vector space at each vertex,
- — a linear transformation on each arrow.

Homomorphisms of representations

$$\begin{tikzcd} R: & V_1 \arrow[r, "f_a"] & V_2 \arrow[r, "f_b"] & V_3 \\ S: & W_1 \arrow[r, "g_a"] & W_2 \arrow[r, "g_b"] & W_3 \end{tikzcd}$$

Homomorphisms of representations

$$\begin{tikzcd}
R\colon \arrow[d, "h"] & V_1 \arrow[r, "f_a"] & V_2 \arrow[r, "f_b"] & V_3 \\
S\colon & W_1 \arrow[r, "g_a"] & W_2 \arrow[r, "g_b"] & W_3
\end{tikzcd}$$

A homomorphism $h\colon R \to S$ is given by:

Homomorphisms of representations

$$\begin{tikzcd}
R\colon \arrow[d, "h"] & V_1 \arrow[r, "f_a"] \arrow[d, "h_1"'] & V_2 \arrow[r, "f_b"] \arrow[d, "h_2"'] & V_3 \arrow[d, "h_3"'] \\
S\colon & W_1 \arrow[r, "g_a"] & W_2 \arrow[r, "g_b"] & W_3
\end{tikzcd}$$

A homomorphism $h\colon R \to S$ is given by:

— linear maps h_i for every vertex i,

Homomorphisms of representations

$$\begin{tikzcd}
R\colon \arrow[d, "h"] & V_1 \arrow[r, "f_a"] \arrow[d, "h_1"'] \arrow[dr, phantom, "\circ"] & V_2 \arrow[r, "f_b"] \arrow[d, "h_2"'] \arrow[dr, phantom, "\circ"] & V_3 \arrow[d, "h_3"'] \\
S\colon & W_1 \arrow[r, "g_a"] & W_2 \arrow[r, "g_b"] & W_3
\end{tikzcd}$$

A homomorphism $h\colon R \to S$ is given by:

- linear maps h_i for every vertex i,
- commuting with the linear maps for the arrows.

Representations and modules

$$\operatorname{mod} kQ \simeq \operatorname{Rep}_k Q$$

Representations and modules

$$\operatorname{mod} kQ \simeq \operatorname{Rep}_k Q$$

finitely generated *kQ*-modules

Representations and modules

$$\operatorname{mod} kQ \simeq \operatorname{Rep}_k Q$$

finitely generated *kQ*-modules

representations of *Q* over *k*

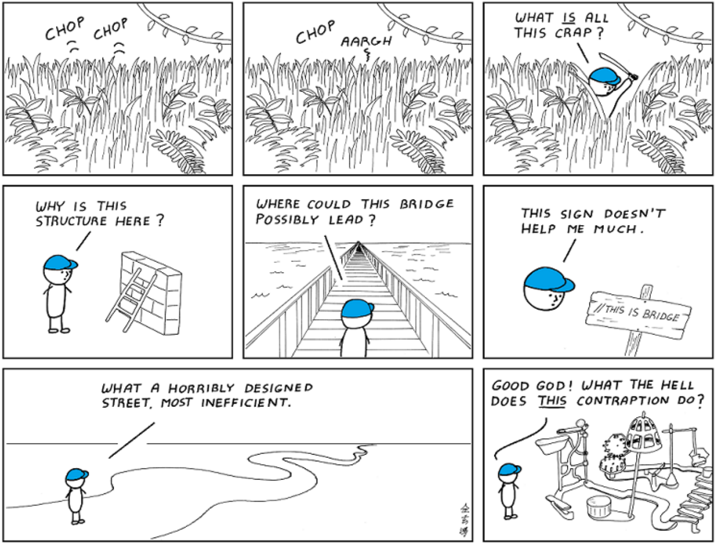
Part 2

A brief history of QPA

A brief history of QPA

- Started by Ed Green at Virginia Tech ca 1998(?)
- Originally called HOPF
- Changed and extended by many people over the years
- Name changed to QPA at some point
- Adopted by Øyvind Solberg (NTNU, Trondheim) in 2010

QPA, ca. 2014



http://www.abstrusegoose.com/432 (CC BY-NC 3.0 US)

What do we do?

```
$
```

What do we do?

```
$ mkdir ~/src/qpa2
$
```

What do we do?

```
$ mkdir ~/src/qpa2
$ cd ~/src/qpa2
$
```

What do we do?

```
$ mkdir ~/src/qpa2
$ cd ~/src/qpa2
$ git init
```

What do we do?

```
$ mkdir ~/src/qpa2
$ cd ~/src/qpa2
$ git init
Initialized empty Git repository in
/home/oysteini/src/qpa2
$
```

What do we do?

```
$ mkdir ∼/src/qpa2
$ cd ∼/src/qpa2
$ git init
Initialized empty Git repository in
/home/oysteini/src/qpa2
$ mkdir lib
$
```

What do we do?

```
$ mkdir ~/src/qpa2
$ cd ~/src/qpa2
$ git init
Initialized empty Git repository in
/home/oysteini/src/qpa2
$ mkdir lib
$ emacs lib/quiver.gd
```

What do we do?

```
$ mkdir ~/src/qpa2
$ cd ~/src/qpa2
$ git init
Initialized empty Git repository in
/home/oysteini/src/qpa2
$ mkdir lib
$ emacs lib/quiver.gd
```

… and so on

QPA version 2

- Complete rewrite of QPA
- Aim: Cleaner and more consistent code
- Started in 2015, still not ready to replace QPA1

Part 3

CAP and us

Some of the problems in QPA1

- Much of QPA is really about certain abelian categories and functors
- But not explicitly: Nothing in GAP lets us say “this is a category” or “this is a functor”
- Have many functions that deal with categorical and functorial aspects, but not in a consistent or complete manner
- Have arbitrarily added (and named) functions that are “too general” – they are applicable in any (abelian) category

Example: Kernel

$$\begin{tikzcd}M \arrow[r, "f"] & N\end{tikzcd}$$

Example: Kernel

$$\operatorname{Ker} f \qquad M \xrightarrow{f} N$$

Example: Kernel

$$\begin{tikzcd}
\operatorname{Ker} f \arrow[r, "\text{inc}"] & M \arrow[r, "f"] & N
\end{tikzcd}$$

Example: Kernel

$$\begin{tikzcd}
\operatorname{Ker} f \arrow[r, "\text{inc}"] & M \arrow[r, "f"] & N \\
X \arrow[ru] & &
\end{tikzcd}$$

Example: Kernel

$$\begin{tikzcd}
\operatorname{Ker} f \arrow[r, "\mathrm{inc}"] & M \arrow[r, "f"] & N \\
X \arrow[u, dashed] \arrow[ur] & &
\end{tikzcd}$$

Example: Kernel

$$\begin{tikzcd}
\operatorname{Ker} f \arrow[r, "\text{inc}"] & M \arrow[r, "f"] & N \\
X \arrow[u, dashed] \arrow[ur] & &
\end{tikzcd}$$

$$\begin{tikzcd}
M \arrow[r, "f"] \arrow[d, "\mu"] & N \arrow[d, "\nu"] \\
M' \arrow[r, "f'"] & N'
\end{tikzcd}$$

Example: Kernel

$$\begin{tikzcd}
\operatorname{Ker} f \arrow[r, "\text{inc}"] & M \arrow[r, "f"] & N \\
X \arrow[u, dashed] \arrow[ur] & &
\end{tikzcd}$$

$$\begin{tikzcd}
\operatorname{Ker} f \arrow[r, "\text{inc}"] & M \arrow[r, "f"] \arrow[d, "\mu"] & N \arrow[d, "\nu"] \\
\operatorname{Ker} f' \arrow[r, "\text{inc}"] & M' \arrow[r, "f'"] & N'
\end{tikzcd}$$

Example: Kernel

$$\begin{tikzcd}
\operatorname{Ker} f \arrow[r, "\text{inc}"] & M \arrow[r, "f"] & N \\
X \arrow[u, dashed] \arrow[ur] & &
\end{tikzcd}$$

$$\begin{tikzcd}
\operatorname{Ker} f \arrow[r, "\text{inc}"] \arrow[d, dashed] & M \arrow[r, "f"] \arrow[d, "\mu"] & N \arrow[d, "\nu"] \\
\operatorname{Ker} f' \arrow[r, "\text{inc}"] & M' \arrow[r, "f'"] & N'
\end{tikzcd}$$

Example: Chain complexes

$$C\colon\ \cdots \to C_2 \xrightarrow{d_2} C_1 \xrightarrow{d_1} C_0 \xrightarrow{d_0} C_{-1} \xrightarrow{d_{-1}} C_{-2} \to \cdots$$

Example: Chain complexes

$$C\colon\ \cdots \to C_2 \xrightarrow{d_2} C_1 \xrightarrow{d_1} C_0 \xrightarrow{d_0} C_{-1} \xrightarrow{d_{-1}} C_{-2} \to \cdots$$

Represented by repeating list (r_1, r_2, r_3) of differentials:

Example: Chain complexes

$$C\colon \; \cdots \to C_2 \xrightarrow{d_2} C_1 \xrightarrow{d_1} C_0 \xrightarrow{d_0} C_{-1} \xrightarrow{d_{-1}} C_{-2} \to \cdots$$

Represented by repeating list (r_1, r_2, r_3) of differentials:

$$\cdots \xrightarrow{d_9} \xrightarrow{d_8} \xrightarrow{d_7} \xrightarrow{d_6} \xrightarrow{d_5} \xrightarrow{d_4} \xrightarrow[r_3]{d_3} \xrightarrow[r_2]{d_2} \xrightarrow[r_1]{d_1}$$

Example: Chain complexes

$$C\colon \cdots \to C_2 \xrightarrow{d_2} C_1 \xrightarrow{d_1} C_0 \xrightarrow{d_0} C_{-1} \xrightarrow{d_{-1}} C_{-2} \to \cdots$$

Represented by repeating list (r_1, r_2, r_3) of differentials:

$$\cdots \xrightarrow{d_9} \xrightarrow{d_8} \xrightarrow{d_7} \xrightarrow[r_3]{d_6} \xrightarrow[r_2]{d_5} \xrightarrow[r_1]{d_4} \xrightarrow[r_3]{d_3} \xrightarrow[r_2]{d_2} \xrightarrow[r_1]{d_1}$$

Example: Chain complexes

$$C\colon \cdots \to C_2 \xrightarrow{d_2} C_1 \xrightarrow{d_1} C_0 \xrightarrow{d_0} C_{-1} \xrightarrow{d_{-1}} C_{-2} \to \cdots$$

Represented by repeating list (r_1, r_2, r_3) of differentials:

$$\cdots \xrightarrow[r_3]{d_9} \xrightarrow[r_2]{d_8} \xrightarrow[r_1]{d_7} \xrightarrow[r_3]{d_6} \xrightarrow[r_2]{d_5} \xrightarrow[r_1]{d_4} \xrightarrow[r_3]{d_3} \xrightarrow[r_2]{d_2} \xrightarrow[r_1]{d_1}$$

Example: Chain complexes

$$C\colon\ \cdots \to C_2 \xrightarrow{d_2} C_1 \xrightarrow{d_1} C_0 \xrightarrow{d_0} C_{-1} \xrightarrow{d_{-1}} C_{-2} \to \cdots$$

Represented by repeating list (r_1, r_2, r_3) of differentials:

$$\cdots \xrightarrow[r_3]{d_9} \xrightarrow[r_2]{d_8} \xrightarrow[r_1]{d_7} \xrightarrow[r_3]{d_6} \xrightarrow[r_2]{d_5} \xrightarrow[r_1]{d_4} \xrightarrow[r_3]{d_3} \xrightarrow[r_2]{d_2} \xrightarrow[r_1]{d_1}$$

Represented by initial differential d_1 and inductive function f:

Example: Chain complexes

$$\begin{tikzcd}C\colon \cdots \arrow[r] & C_2 \arrow[r, "d_2"] & C_1 \arrow[r, "d_1"] & C_0 \arrow[r, "d_0"] & C_{-1} \arrow[r, "d_{-1}"] & C_{-2} \arrow[r] & \cdots\end{tikzcd}$$

Represented by repeating list (r_1, r_2, r_3) of differentials:

$$\begin{tikzcd}\cdots \arrow[r, "d_9", "r_3"'] & {} \arrow[r, "d_8", "r_2"'] & {} \arrow[r, "d_7", "r_1"'] & {} \arrow[r, "d_6", "r_3"'] & {} \arrow[r, "d_5", "r_2"'] & {} \arrow[r, "d_4", "r_1"'] & {} \arrow[r, "d_3", "r_3"'] & {} \arrow[r, "d_2", "r_2"'] & {} \arrow[r, "d_1", "r_1"'] & {}\end{tikzcd}$$

Represented by initial differential d_1 and inductive function f:

$$\begin{tikzcd}\cdots \arrow[r] & {} \arrow[r] & {} \arrow[r] & {} \arrow[r, "d_1"] & {}\end{tikzcd}$$

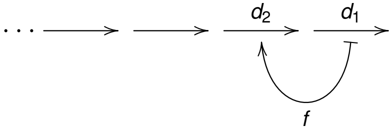
Example: Chain complexes

$$C\colon \cdots \to C_2 \xrightarrow{d_2} C_1 \xrightarrow{d_1} C_0 \xrightarrow{d_0} C_{-1} \xrightarrow{d_{-1}} C_{-2} \to \cdots$$

Represented by repeating list (r_1, r_2, r_3) of differentials:

$$\cdots \xrightarrow[r_3]{d_9} \xrightarrow[r_2]{d_8} \xrightarrow[r_1]{d_7} \xrightarrow[r_3]{d_6} \xrightarrow[r_2]{d_5} \xrightarrow[r_1]{d_4} \xrightarrow[r_3]{d_3} \xrightarrow[r_2]{d_2} \xrightarrow[r_1]{d_1}$$

Represented by initial differential d_1 and inductive function f:



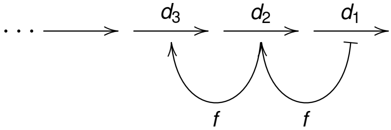
Example: Chain complexes

$$C\colon \cdots \to C_2 \xrightarrow{d_2} C_1 \xrightarrow{d_1} C_0 \xrightarrow{d_0} C_{-1} \xrightarrow{d_{-1}} C_{-2} \to \cdots$$

Represented by repeating list (r_1, r_2, r_3) of differentials:

$$\begin{tikzcd}
\cdots \arrow[r, "d_9", "r_3"'] & {} & {} \arrow[r, "d_8", "r_2"'] & {} & {} \arrow[r, "d_7", "r_1"'] & {} & {} \arrow[r, "d_6", "r_3"'] & {} & {} \arrow[r, "d_5", "r_2"'] & {} & {} \arrow[r, "d_4", "r_1"'] & {} & {} \arrow[r, "d_3", "r_3"'] & {} & {} \arrow[r, "d_2", "r_2"'] & {} & {} \arrow[r, "d_1", "r_1"'] & {}
\end{tikzcd}$$

Represented by initial differential d_1 and inductive function f:



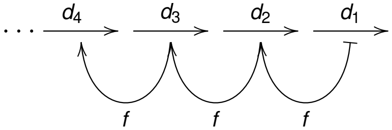
Example: Chain complexes

$$C\colon \cdots \to C_2 \xrightarrow{d_2} C_1 \xrightarrow{d_1} C_0 \xrightarrow{d_0} C_{-1} \xrightarrow{d_{-1}} C_{-2} \to \cdots$$

Represented by repeating list (r_1, r_2, r_3) of differentials:

$$\cdots \xrightarrow[r_3]{d_9} \; \xrightarrow[r_2]{d_8} \; \xrightarrow[r_1]{d_7} \; \xrightarrow[r_3]{d_6} \; \xrightarrow[r_2]{d_5} \; \xrightarrow[r_1]{d_4} \; \xrightarrow[r_3]{d_3} \; \xrightarrow[r_2]{d_2} \; \xrightarrow[r_1]{d_1}$$

Represented by initial differential d_1 and inductive function f:



Status QPA project early 2015

Wanted to:

- Rewrite QPA in a cleaner and nicer way
- Move functionality for chain complexes into a separate package

Status QPA project early 2015

Wanted to:

- Rewrite QPA in a cleaner and nicer way
- Move functionality for chain complexes into a separate package

One big missing piece:

- General structures for categories and functors

Status QPA project early 2015

Wanted to:

- Rewrite QPA in a cleaner and nicer way
- Move functionality for chain complexes into a separate package

One big missing piece:

- General structures for categories and functors

… and then we learned about CAP.

Result

- QPA2 is written with CAP in mind (almost) from the beginning

Categories in QPA2

Categories in QPA2

vec *k*

Categories in QPA2

$$\begin{tikzcd}
\operatorname{Rep}_k Q \\
\operatorname{vec} k \arrow[u, squiggly]
\end{tikzcd}$$

Categories in QPA2

$$\begin{tikzcd}
\mathrm{Rep}_k\, Q \\
\mathrm{vec}\, k \arrow[u, squiggly]
\end{tikzcd}$$

$$\begin{tikzcd}
k^3 \arrow[r] & k \arrow[r] & k^2
\end{tikzcd}$$

Categories in QPA2

$$\begin{tikzcd}
\operatorname{mod} kQ \\
\operatorname{Rep}_k Q \arrow[u, squiggly] \\
\operatorname{vec} k \arrow[u, squiggly]
\end{tikzcd}$$

$$\begin{tikzcd}
k^3 \arrow[r] & k \arrow[r] & k^2
\end{tikzcd}$$

Categories in QPA2

$$\begin{tikzcd}
\mathrm{mod}\, kQ \\
\mathrm{Rep}_k\, Q \arrow[u, squiggly] \\
\mathrm{vec}\, k \arrow[u, squiggly]
\end{tikzcd}$$

$$\begin{tikzcd}
 & M & \\
k^3 \arrow[r] & k \arrow[r] & k^2
\end{tikzcd}$$

Functors in QPA2: Hom and tensor (work in progress)

$$\mathrm{Hom}_A({}_A M, -)\colon A\text{-mod} \to \mathrm{vec}\, k$$

Functors in QPA2: Hom and tensor (work in progress)

$$\operatorname{Hom}_A({}_AM, -)\colon A\text{-mod} \to \operatorname{vec} k$$
$$\operatorname{Hom}_A({}_AM_B, -)\colon A\text{-mod} \to B\text{-mod}$$

Functors in QPA2: Hom and tensor (work in progress)

$$\begin{aligned}
\operatorname{Hom}_A({}_AM, -)\colon A\text{-mod} &\to \operatorname{vec} k \\
\operatorname{Hom}_A({}_AM_B, -)\colon A\text{-mod} &\to B\text{-mod} \\
\operatorname{Hom}_A({}_AM_B, -)\colon A\text{-mod-}C &\to B\text{-mod-}C
\end{aligned}$$

Functors in QPA2: Hom and tensor (work in progress)

$$\begin{aligned}
\operatorname{Hom}_A({}_AM, -)\colon A\text{-mod} &\to \operatorname{vec} k\\
\operatorname{Hom}_A({}_AM_B, -)\colon A\text{-mod} &\to B\text{-mod}\\
\operatorname{Hom}_A({}_AM_B, -)\colon A\text{-mod-}C &\to B\text{-mod-}C
\end{aligned}$$

$$M_A \otimes_A - \colon A\text{-mod} \to \operatorname{vec} k$$

Functors in QPA2: Hom and tensor (work in progress)

$$\mathrm{Hom}_A({}_AM, -)\colon A\text{-mod} \to \mathrm{vec}\, k$$
$$\mathrm{Hom}_A({}_AM_B, -)\colon A\text{-mod} \to B\text{-mod}$$
$$\mathrm{Hom}_A({}_AM_B, -)\colon A\text{-mod-}C \to B\text{-mod-}C$$

$$M_A \otimes_A - \colon A\text{-mod} \to \mathrm{vec}\, k$$
$${}_BM_A \otimes_A - \colon A\text{-mod} \to B\text{-mod}$$

Functors in QPA2: Hom and tensor (work in progress)

$$\operatorname{Hom}_A({}_AM, -)\colon A\text{-mod} \to \operatorname{vec} k$$
$$\operatorname{Hom}_A({}_AM_B, -)\colon A\text{-mod} \to B\text{-mod}$$
$$\operatorname{Hom}_A({}_AM_B, -)\colon A\text{-mod-}C \to B\text{-mod-}C$$

$$M_A \otimes_A -\colon A\text{-mod} \to \operatorname{vec} k$$
$${}_BM_A \otimes_A -\colon A\text{-mod} \to B\text{-mod}$$
$${}_BM_A \otimes_A -\colon A\text{-mod-}C \to B\text{-mod-}C$$

Main advantages of using CAP (for us)

Main advantages of using CAP (for us)

- CAP gives us a consistent and complete interface for (abelian) categories.

Main advantages of using CAP (for us)

- CAP gives us a consistent and complete interface for (abelian) categories.
- Clear separation between general things that can be done in any abelian category and the things that are specific to our categories

Main advantages of using CAP (for us)

- CAP gives us a consistent and complete interface for (abelian) categories.
- Clear separation between general things that can be done in any abelian category and the things that are specific to our categories
- Categories and functors are objects

Main advantages of using CAP (for us)

- CAP gives us a consistent and complete interface for (abelian) categories.
- Clear separation between general things that can be done in any abelian category and the things that are specific to our categories
- Categories and functors are objects
- CAP makes our code better structured

Possible disadvantages

Possible disadvantages

- CAP is big and complex

Possible disadvantages

- CAP is big and complex
- Deeply nested function calls complicate debugging

Possible disadvantages

- CAP is big and complex
- Deeply nested function calls complicate debugging
- Heavy use of CAP could slow down the code

Possible disadvantages

— CAP is big and complex

— Deeply nested function calls complicate debugging

— Heavy use of CAP could slow down the code

— Confusing terminology: “GAP categories” and “CAP categories”

Possible disadvantages: Response

- *CAP is big and complex*
- *Deeply nested function calls complicate debugging*
- *Heavy use of CAP could slow down the code*
- *Confusing terminology: “GAP categories” and “CAP categories”*

Possible disadvantages: Response

— *CAP is big and complex*

. . . but still reasonably easy to use in simple ways

— *Deeply nested function calls complicate debugging*

— *Heavy use of CAP could slow down the code*

— *Confusing terminology: “GAP categories” and “CAP categories”*

Possible disadvantages: Response

— *CAP is big and complex*
. . . but still reasonably easy to use in simple ways

— *Deeply nested function calls complicate debugging*
. . . but that is probably necessary

— *Heavy use of CAP could slow down the code*

— *Confusing terminology: “GAP categories” and “CAP categories”*

Possible disadvantages: Response

— *CAP is big and complex*
. . . but still reasonably easy to use in simple ways

— *Deeply nested function calls complicate debugging*
. . . but that is probably necessary

— *Heavy use of CAP could slow down the code*
. . . but there might be workarounds, such as turning off caching

— *Confusing terminology: “GAP categories” and “CAP categories”*

Possible disadvantages: Response

— *CAP is big and complex*
. . . but still reasonably easy to use in simple ways

— *Deeply nested function calls complicate debugging*
. . . but that is probably necessary

— *Heavy use of CAP could slow down the code*
. . . but there might be workarounds, such as turning off caching

— *Confusing terminology: "GAP categories" and "CAP categories"*
. . . but that is GAP's fault, not CAP's

Wish list

Wish list

- Parametrized categories

Wish list

- Parametrized categories: mod $*$

Wish list

- Parametrized categories: $\operatorname{mod} * \rightsquigarrow \operatorname{mod} A$

Wish list

- Parametrized categories: $\operatorname{mod} * \rightsquigarrow \operatorname{mod} A$
- Parametrized functors

Wish list

- Parametrized categories: $\mathrm{mod}\, * \rightsquigarrow \mathrm{mod}\, A$
- Parametrized functors: $\mathrm{Hom}_*(*, -)$

Wish list

- Parametrized categories: $\operatorname{mod} * \rightsquigarrow \operatorname{mod} A$
- Parametrized functors: $\operatorname{Hom}_*(*, -) \rightsquigarrow \operatorname{Hom}_A(M, -)$

Wish list

- Parametrized categories: $\operatorname{mod} * \rightsquigarrow \operatorname{mod} A$
- Parametrized functors: $\operatorname{Hom}_*(*, -) \rightsquigarrow \operatorname{Hom}_A(M, -)$
- Contravariant functors

Wish list

- Parametrized categories: mod $*$ $\rightsquigarrow$ mod A
- Parametrized functors: $\mathrm{Hom}_*(*, -)$ $\rightsquigarrow$ $\mathrm{Hom}_A(M, -)$
- Contravariant functors
- Functors with two arguments: $\mathrm{Hom}(-, -)$

Wish list

- Parametrized categories: mod $*$ $\rightsquigarrow$ mod A
- Parametrized functors: $\mathrm{Hom}_*(*, -) \rightsquigarrow \mathrm{Hom}_A(M, -)$
- Contravariant functors
- Functors with two arguments: $\mathrm{Hom}(-, -)$
- CAP should,

Wish list

- Parametrized categories: mod $* \rightsquigarrow$ mod A
- Parametrized functors: $\mathrm{Hom}_*(*, -) \rightsquigarrow \mathrm{Hom}_A(M, -)$
- Contravariant functors
- Functors with two arguments: $\mathrm{Hom}(-, -)$
- CAP should, like all the best free software projects,

Wish list

- Parametrized categories: mod $*$ $\rightsquigarrow$ mod A
- Parametrized functors: $\mathrm{Hom}_*(*, -)$ $\rightsquigarrow$ $\mathrm{Hom}_A(M, -)$
- Contravariant functors
- Functors with two arguments: $\mathrm{Hom}(-, -)$
- CAP should, like all the best free software projects, have a cute animal as mascot.

Wish list

- Parametrized categories: mod $*$ $\rightsquigarrow$ mod A
- Parametrized functors: $\mathrm{Hom}_*(*, -) \rightsquigarrow \mathrm{Hom}_A(M, -)$
- Contravariant functors
- Functors with two arguments: $\mathrm{Hom}(-, -)$
- CAP should, like all the best free software projects,
  have a cute animal as mascot.
  Suggestion:

Wish list

- Parametrized categories: mod $*$ $\rightsquigarrow$ mod A
- Parametrized functors: $\mathrm{Hom}_*(*, -)$ $\rightsquigarrow$ $\mathrm{Hom}_A(M, -)$
- Contravariant functors
- Functors with two arguments: $\mathrm{Hom}(-, -)$
- CAP should, like all the best free software projects,
  have a cute animal as mascot.
  Suggestion: A cat

Wish list

- Parametrized categories: $\operatorname{mod} * \rightsquigarrow \operatorname{mod} A$
- Parametrized functors: $\operatorname{Hom}_*(*, -) \rightsquigarrow \operatorname{Hom}_A(M, -)$
- Contravariant functors
- Functors with two arguments: $\operatorname{Hom}(-, -)$
- CAP should, like all the best free software projects,
  have a cute animal as mascot.
  Suggestion: A cat wearing a cap

Wish list

- Parametrized categories: mod $* \rightsquigarrow$ mod A
- Parametrized functors: $\mathrm{Hom}_*(*, -) \rightsquigarrow \mathrm{Hom}_A(M, -)$
- Contravariant functors
- Functors with two arguments: $\mathrm{Hom}(-, -)$
- CAP should, like all the best free software projects,
  have a cute animal as mascot.
  Suggestion: A cat wearing a cap with the text “CAP”