Representations of a concrete category as objects in the functor category

Tibor Grün

August 7, 2020

Contents

1 Introduction

$$\mathbf{Quiv} \to^{CatClosure} \leftarrow_{U} \mathbf{Cats} \to^{k-Algebroid} \leftarrow_{U} \mathbf{k}-\mathbf{Cats} \to^{AdditiveClosure} \leftarrow_{U} \mathbf{k}-\mathbf{Cats}^{\oplus}$$

2 A short overview of the tools used

GAP, QPA / QPA2, Catreps, CAP, homalg_project

3 Introduction in category theory

This section serves two purposes: On the one hand, it is an introduction to quivers and category theory. On the other hand it introduces concrete categories which we want to represent, and all the additional constructions that are needed to that goal.

3.1 Quivers

In this section, we first want to define the category **Quiv** and how it is the prototype for the category **Cats**. In order to describe the category **Quiv** of quivers, we first have to define what a category is and for this we need the definition of a quiver. Lateron we will revisit this definition as we can define quivers as the objects in the quiver category **Quiv**.

Definition 3.1.1. (Quiver)
A directed graph or quiver q consists of a class of objects (or vertices) $q_0 = \operatorname{Obj} q$ and a class of morphisms (or arrows) $q_1 = \operatorname{Mor} q$ together with two defining maps

$$s, t \colon q_1 \rightrightarrows q_0$$

s called source and t called target.

In the next definition we are giving a new characterization for q_1 by looking at all arrows between two fixed objects.

Definition 3.1.2. (Hom-set of a (locally) small quiver)

(1) Given two objects $M, N \in q_0$ we write $\operatorname{Hom}_q(M,N)$ or $q(M,N)$ for the fiber $(s,t)^{-1}(\{(M,N)\})$ of the product map $(s,t) : q_1 \longrightarrow q_0 \times q_0$ over the pair $(M,N) \in q_0 \times q_0$. This is the class of all morphisms with source $= M$ and target $= N$. We indicate this by writing $\varphi : M \longrightarrow N$ or $M \xrightarrow{\varphi} N$. Hence q_1 is the disjoint union $\dot{\bigcup}_{M,N \in q_0} \operatorname{Hom}_q(M,N) = q_1$. As usual we define $\operatorname{End}_q(M) := \operatorname{Hom}_q(M,M)$.

(2) If the class $\operatorname{Hom}_q(M,N)$ is a set for all pairs (M,N) then we call the quiver locally small. We therefore talk about Hom-sets. If additionally, q_0 is a set, then the quiver is called small.

(3) A quiver with a finite set of objects and a finite set of morphisms is called a finite quiver.

Example 3.1.3. (Quiver with 2 objects and 3 morphisms)

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

The objects of this quiver q are $q_0 = \{1,2\}$, and the morphisms are $q_1 = \{a,b,c\}$ with
$s(a) = 1 = t(a)$, $s(c) = 2 = t(c)$ and $s(b) = 1, t(b) = 2$.
Thus $\operatorname{End}_q(1) = \{a\}, \operatorname{End}_q(2) = \{c\}$ and $\operatorname{Hom}_q(1,2) = \{b\}$ whereas $\operatorname{Hom}_q(2,1) = \varnothing$.

In `QPA` this quiver is encoded as `q(2)[a:1->1,b:1->2,c:2->2]` where the first `(2)` in parentheses stands for the total number of objects.

Definition 3.1.4. (Composable arrows; path in a quiver)
Since we already have the source and target maps, we say two arrows $a, b \in q_1$ are composable if $t(a) = s(b)$ or $t(b) = s(a)$. In this case we can write a sequence of composable arrows $p = a_1 a_2 \cdots a_n$ where $t(a_i) = s(a_{i+1})$ for $i = 1, \ldots, n-1$. We call this sequence a path from $s(a_1)$ to $t(a_n)$ and the integer $n \in \mathbb{Z}_{\geq 0}$ the length $l(p)$ of the path p. Although it's not an arrow, we can define the source and target of a path $p = a_1 \cdots a_n$ as $s(p) := s(a_1)$ and $t(p) := t(a_n)$. A path $p = a_1 \cdots a_n$ with $s(a_1) = t(a_n)$, i.e. $s(p) = t(p)$, is called cyclic.
For an endomorphism $a \in \mathrm{End}_q(M)$ we write a^n for $aa \cdots a$ (n times). In the case of $n = 0$ an empty path whose source and target are the vertex $i \in q_0$ is called the trivial path at i and is denoted e_i. Note that the composition of paths $e_i e_i$ has length zero starting at i therefore $e_i^2 = e_i$.

Lemma 3.1.5. *Let Q be a quiver. If there is a path of length at least $|Q_0|$, then there are cyclic paths, and thus infinitely many paths.[2]*

Proof. Assume that there exists a path of length greater or equal to $|Q_0|$. Then there exists a path of length $n = |Q_0|$, say $\alpha_1 \cdots \alpha_n$. Consider the vertices $x_i = s(\alpha_i)$ for $1 \leq i \leq n$ and $x_{n+1} = t(\alpha_n)$. Then these are $n+1$ vertices, thus there has to exist $i < j$ with $x_i = x_j$. Let $\omega = \alpha_i \cdots \alpha_{j-1}$, this is a path with source and target $x_i = x_j$, thus a cyclic path. But then ω^m is a path for any natural number m. The path ω has length $j - i \geq 1$, thus ω^m has length $m(j-i)$. This shows that these paths are pairwise different. □

Example 3.1.6. (A quiver with no cycles)

$$\begin{tikzcd}
2 \arrow[rr, "\psi"] \arrow[ddrr, "\psi\rho" near start] & & 3 \arrow[dd, "\rho"] \\
& & \\
1 \arrow[uu, "\varphi"] \arrow[uurr, "\varphi\psi" near start] \arrow[rr, "\varphi\psi\rho"'] & & 4
\end{tikzcd}$$

The longest path $1 \to 2 \to 3 \to 4$ has length 3. If after the object 4 another arrow would go to either $1, 2, 3$ or 4 itself, we would have a cyclic path and thus infinitely many paths.

3.2 Categories

Definition 3.2.1. (Category)
A category $\mathcal{C}$ is a quiver with two further maps:

(id) The identity map $1_{()}$ mapping every object $X \in \mathcal{C}_0$ to its identity morphism 1_X:

$$\mathcal{C}_0 \xrightarrow{1} \mathcal{C}_1$$

(μ) And for any two composable morphisms φ and $\psi \in \mathcal{C}_1$, i.e. with $t(\varphi) = s(\psi)$, the composition map μ, which maps $\varphi, \psi \in \mathcal{C}_1 \times \mathcal{C}_1$ to $\mu(\varphi, \psi) \in \mathcal{C}_1$ which we also write as $\varphi\psi$.

$$\mathcal{C}_1 \times \mathcal{C}_1 \xrightarrow{\mu} \mathcal{C}_1$$

The defining properties for 1 and μ are:

(1) $s(1_M) = M = t(1_M)$, i.e.
$1_M \in \mathrm{End}_{\mathcal{C}} \, \forall M \in \mathcal{C}$.

(2) $s(\varphi\psi) = s(\varphi)$ and
$t(\varphi\psi) = t(\psi)$
for all composable morphisms $\varphi, \psi \in \mathcal{C}$.

$$\mu : \mathrm{Hom}_{\mathcal{C}}(M, L) \times \mathrm{Hom}_{\mathcal{C}}(L, N) \longrightarrow \mathrm{Hom}_{\mathcal{C}}(M, N)$$

(3) $(\varphi\psi)\rho = \varphi(\psi\rho)$ [associativity of composition]

(4) $1_{s(\varphi)}\varphi = \varphi = \varphi 1_{t(\varphi)}$ [unit property]
The identity is a left and right unit of the composition.

3.3 Functors

Categories are themselves objects in the category of categories, which leads to a question: What is a morphism between categories?

Definition 3.3.1. (Functor)
A functor $F : \mathcal{C} \to \mathcal{D}$, between categories $\mathcal{C}$ and $\mathcal{D}$, consists of the following data:

- An object $Fc \in \mathcal{D}_0$, for each object $c \in \mathcal{C}_0$.
- A morphism $Ff : Fc \to Fc' \in \mathcal{D}_1$, for each morphism $f : c \to c' \in \mathcal{C}_1$, so that the domain and codomain of Ff are, respectively, equal to F applied to the domain or codomain of f.

The assignments are required to satisfy the following two functoriality axioms:

- For any composable pair $f, g \in \mathcal{C}_1, Fg \cdot Ff = F(g \cdot f)$.
- For each object $c \in \mathcal{C}_0, F(1_c) = 1_{Fc}$.

Put concisely, a functor consists of a mapping on objects and a mapping on morphisms that preserves all of the structure of a category, namely domains and codomains, composition, and identities.

3.4 Natural transformations

With fixed categories $\mathcal{C}$ and $\mathcal{D}$ we can consider functors $F, G \in \mathrm{Hom}(\mathcal{C}, \mathcal{D})$ themselves as objects in the category $\mathrm{Hom}(\mathcal{C}, \mathcal{D})$ of functors between $\mathcal{C}$ and $\mathcal{D}$. In this functor category, the morphisms between two functors are called natural transformations.

Definition 3.4.1. (Natural transformations)
Given categories $\mathcal{C}$ and $\mathcal{D}$ and functors $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{C} \to \mathcal{D}$, a natural transformation $\alpha : F \Rightarrow G$ consists of:

- an arrow $\alpha_c : Fc \to Gc \in \mathcal{D}_1$ for each object $c \in \mathcal{C}_0$, the collection of which define the components of the natural transformation, so that, for any morphism $f : c \to c' \in \mathcal{C}_1$, the following square of morphisms in $\mathcal{D}$

$$\begin{tikzcd} Fc \arrow[r, "\alpha_c"] \arrow[d, "Ff"'] & Gc \arrow[d, "Gf"] \\ Fc' \arrow[r, "\alpha_{c'}"] & Gc' \end{tikzcd}$$

  commutes, i.e., has a a common composite $Fc \to Gc' \in \mathcal{D}_1$.

4 Finite concrete categories

Definition 4.0.1. (Finite and concrete categories)

(1) A finite category is a category with a finite set of objects and a finite set of morphisms.

(2) A concrete category is a category whose objects have underlying sets (or are themselves sets) and whose morphisms are functions between these underlying sets. Otherwise it's called an abstract category.

Clearly every finite concrete category is a small category.

Remark 4.0.2 (Implementation)**.** Using the implementation of `FinSets` in we implement a finite concrete category as a subcategory of FinSets. The finite concrete category is generated by its Set Of Generating Morphisms $\{g_1, \ldots, g_r\}$.

When our goal is representation of finite concrete categories, i.e. functors $\mathcal{C} \to k$-Mat, why are we not defining the functor $\mathrm{Hom}(\mathcal{C}, k\text{-Mat})$ but instead first define the k-Algebroid kq of the `RightQuiverFromConcreteCategory`$(\mathcal{C})$ and then the functor $\mathrm{Hom}(kq, k\text{-Mat})$?

Definition 4.0.3. A congruence relation $\sim$ on a category $\mathcal{C}$ is an equivalence relation on the set of morphisms $\mathcal{C}_1$ such that for pre-composable c and post-composable $d \in \mathcal{C}_1$:

$$a \sim b \Rightarrow ca \sim cb \wedge ad \sim bd$$

The equivalence classes $[f]_\sim$ again form a set of morphisms $\mathcal{C}_1/\sim$.

We can calculate the `RightQuiverFromConcreteCategory`. We can calculate the `CategoryClosure` of that quiver indirectly by first calculating the `Algebroid`$(k,\mathcal{C})$ and then the `UnderlyingCategory`.

As a subcategory of **FinSets**, our finite concrete category $\mathcal{C}$ does not have a pre-additive structure on it, i.e. for two objects $M, N, \mathrm{Hom}_{\mathcal{C}}(M,N)$ does not have the structure of an abelian group.

The category **FinSets** as a subcategory of **Sets** does not have a zero object, since the empty set $\emptyset$ is the unique initial object and every singleton is a terminal object which is different from the initial object.

Example 4.0.4. Forgetful functor / Category closure / k-Algebroid

When we want to calculate representations of our finite concrete categories, we make the Hom functor Hom(ccat, kMat). But functors from the concrete category directly are not useful when we know nothing about the relations of morphisms in our category. Instead we go an indirect route, first calculating the underlying quiver and from this the k-algebroid, i.e. the path algebra with the endomorphism relations and such.

Algorithm 1: `RightQuiverFromConcreteCategory`

Input : a finite concrete category C with n objects
Output : the right quiver $q(n)$

let Obj be the set of objects of C;
let $n := Length(Obj)$;
let $gMor$ be the set of generating morphisms of C;
let A be the empty set and let $i := 1$;
foreach *morphism mor in gMor* **do**
  let $A_{i,1}$ be the position of $Source(mor)$ in Obj;
  let $A_{i,2}$ be the position of $Range(mor)$ in Obj;
  let $i := i + 1$;
end
let q be the right quiver with vertices $\{1, \ldots, n\}$ and arrows A.
return q;

5 The categories Quiv, Cat, FinSets, k-Mat, CatReps and the Functor Category

In this section, we first want to define the category **Quiv** and how it is the prototype for the category **Cats**. We want to restrict ourselves to finite concrete categories, which brings us to the category **FinSets**. Our goal is to represent concrete categories, for this we need the source and target categories of our representations. The source category is **k-Algebroids** which we compute algorithmically from a concrete category. The target category of our category representations is **k-Mat**. The category where our category representations lie in is **CatReps** for which we show that it's a subcategory of the **Functor Category**.

5.1 Additional structure on the Hom-set of a category

Example 5.1.1. A group G defines a category BG with a single object. The group elements are its morphisms, which are all automorphisms of the single object. The identity element $e \in G$ acts as the identity morphism for the unique object in this category. The hom-set of that category is itself a group.

This example can be generalized to categories where the hom-set is a ring or an R-algebra. But for this we need a commutative ring R and thus the category R-Mod.

Definition 5.1.2. Ab-Category An Ab-category is a category in which all homomorphism sets are abelian groups. That means in addition to the composition of morphisms $\mu : \mathcal{C}_1 \times \mathcal{C}_1 \longrightarrow \mathcal{C}_1$ we have another binary operation $+ : \mathcal{C}_1 \times \mathcal{C}_1 \longrightarrow \mathcal{C}_1$, that distributes over the composition, i.e.

$$\mu(f+g,h) = \mu(f,h) + \mu(g,h)$$

.

The concept of a functor is central in category theory. It is how the objects and morphisms of two categories relate to one another.

Definition 5.1.3. (Functor)
In the category **Cat** which has categories as objects, functors are the morphisms between these objects. Let $\mathcal{C}, \mathcal{D} \in \operatorname{Obj}\mathbf{Cat}$ be categories. A functor $F : \mathcal{C} \longrightarrow \mathcal{D}$ between $\mathcal{C}$ and $\mathcal{D}$ consists of the following data:

(1) For every object $c \in \mathcal{C}_0$ there is an object $Fc \in \mathcal{D}$.

(2) For every morphism $f \in \mathcal{C}_1$, i.e. $c \xrightarrow{f} c'$ there is a morphism $Ff \in \mathcal{D}_1$ with $Fc \xrightarrow{Ff} Fc'$, i.e. $s(Ff) = F\,s(f)$ and $t(Ff) = F\,t(f)$

Functors are compatible with the identity map and the composition map:

(3) For every object $c \in \mathcal{C}_0$ we have $1_c \in \mathcal{C}_1$ and for the functor F we demand that $F1_c = 1_{Fc} \in \mathcal{D}_1$.

(4) For every pair of morphisms $f, g \in \mathcal{C}_1$ with $t(f) = s(g)$ we have $fg \in \mathcal{C}_1$ and we demand that $Fg\,Fg = F\,gf$.

With the definition of a category and the category of functors finished, we can come back and use them to define the category of quivers **Quiv**.

Definition 5.1.4. (The category **Quiv**)[3] Let the Kronecker category $\mathcal{K}$ be the category with two objects, 0 and 1, and two non-identity morphisms, s and t $1 \overset{s}{\underset{t}{\rightrightarrows}} 0$. Let **FinSets** be the category of finite sets with morphisms being maps between those sets. The category of quivers **Quiv** is the category of functors from $\mathcal{K}$ to **FinSets**. For a quiver $q \in \operatorname{Obj}\mathbf{Quiv}$ we write q_x for the image of $x \in \{0,1\}$ under q. The images under q of the morphisms s and t are again denoted by s and t.

As we have seen, every category is a quiver, but in general, to become a category, a quiver is lacking identity morphisms and the composition of morphisms. To be more precise, there is a functor U from the category of categories CAT to the category of quivers Quiv, called the underlying quiver or forgetful functor.

$$\text{Cat} \xrightarrow{U} \text{Quiv}$$

mapping every object $M \in \mathcal{C}_0$ to the same objects in q_0, mapping every arrow $\varphi \in \mathcal{C}_1$ to an arrow $a \in q_1$, respecting source and target, but forgetting the special role of the identity morphisms and of the composition morphisms.

Example 5.1.5. (Underlying quiver)

$$
\begin{tikzcd}
B \arrow[r, "\psi"] \arrow[rd, "\psi\rho" near start] \arrow[loop above, "1_B"] & C \arrow[d, "\rho"] \arrow[loop above, "1_C"] \\
A \arrow[u, "\varphi"] \arrow[ru, "\varphi\psi"' near start] \arrow[r, bend left, "(\varphi\psi)\rho"] \arrow[r, bend right, "\varphi(\psi\rho)"'] \arrow[loop below, "1_A"] & D \arrow[loop below, "1_D"]
\end{tikzcd}
\quad \xRightarrow{U} \quad
\begin{tikzcd}
2 \arrow[r, "b"] \arrow[rd, "e" near start] \arrow[loop above, "h"] & 3 \arrow[d, "c"] \arrow[loop above, "i"] \\
1 \arrow[u, "a"] \arrow[ru, "d"' near start] \arrow[r, bend left, "f"] \arrow[r, bend right, "f"'] \arrow[loop below, "g"] & 4 \arrow[loop below, "j"]
\end{tikzcd}
$$

In the category on the left, associativity of composition guaranteed that $(\varphi\psi)\rho = \varphi(\psi\rho)$, so those two arrows were already the same, so they are mapped to the same arrow $f = U((\varphi\psi)\rho) = U(\varphi(\psi\rho))$ in the quiver on the right. We didn't have to draw both arrows for f, but since they are equal, there is still only one arrow in the hom-set $\mathrm{Hom}_q(1,4) = \{f, f\} = \{f\}$.
All the other identities are not preserved under the forgetful functor, e.g. d doesn't know what it has to do with a and b apart from $s(d) = s(a)$ and $t(d) = t(b)$. Especially the former identity arrows are now just endomorphisms with no defining property.
The paths $g^2 f, gf$ and fj^3 are all different, while in the category, they all simplify to $1_A 1_A(\varphi\psi)\rho = 1_A(\varphi\psi)\rho = (\varphi\psi)\rho 1_D 1_D 1_D = (\varphi\psi)\rho$ due to the unit property and associativity.

Definition 5.1.6. (Ab-category) An <u>Ab-category</u> is a category in which all homomorphism sets are abelian groups, and composition distributes over addition.
In other words, A category $\mathcal{C}$ is an <u>Ab-category</u> if for every pair of objects $M, N \in \mathcal{C}_0$, $(\mathrm{Hom}_{\mathcal{C}}(M,N), +)$ is an abelian group (with the neutral element called <u>zero morphism</u>), and for all morphisms $\gamma, \delta \in \mathrm{Hom}_{\mathcal{C}}(M,N), \alpha, \beta \in \mathrm{Hom}_{\mathcal{C}}(N,L)$

$$(\gamma + \delta)\alpha = \gamma\alpha + \delta\alpha \text{ and}$$

$$\gamma(\alpha + \beta) = \gamma\alpha + \gamma\beta.$$

Note that every hom-set has its own unique zero morphism. E.g. in $\mathrm{Mat}_{\mathbb{Q}}$ the 2-by-3 zero-matrix $0 \in \mathrm{Hom}(2,3)$ is different from the 4-by-4 zero-matrix $0 \in \mathrm{Hom}(4,4)$.

Definition 5.1.7. (Initial object, terminal object, zero object)

Example 5.1.8.

Definition 5.1.9. (Kernel of a morphism

Definition 5.1.10. (Abelian category)

Definition 5.1.11. (k-linear category)

Quiver -¿ CAT: U: forget 1, forget composition
search U^{-1}
Beispiel für Adjunktion
Path Algebra:

6 Functors and natural transformations

6.1 Functors map one category to another

Example 6.1.1. (Identity Functor) bla

Example 6.1.2. (Forgetful functor) bla

Definition 6.1.3. (full functor; faithful functor)

6.2 Natural transformations are morphisms between functors

7 Yoneda's Lemma: Completion and cocompletion of a category

7.1 Embedding categories

Lemma 7.1.1. *(Yoneda's Lemma)*

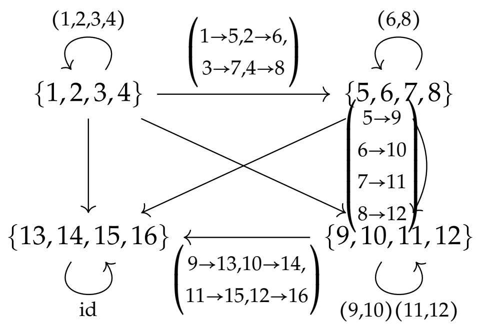
Proof. □

Projective objects?

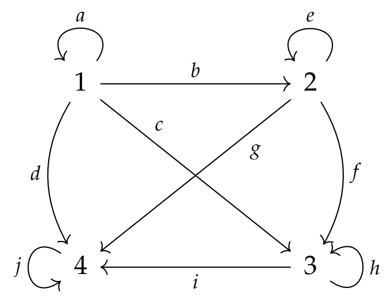
$$\begin{pmatrix}1 & 2 & 3 & 4\end{pmatrix}\begin{pmatrix}1\to 5, & 2\to 6\\ 3\to 7, & 4\to 8\end{pmatrix}\begin{pmatrix}6 & 8\end{pmatrix}\begin{pmatrix}5\to 9\\ 6\to 10\\ 7\to 11\\ 8\to 12\end{pmatrix}\begin{pmatrix}9 & 10\end{pmatrix}\begin{pmatrix}11 & 12\end{pmatrix}\begin{pmatrix}9\to 13, & 10\to 14\\ 11\to 15 & 12\to 16\end{pmatrix}\mathrm{id}$$

7.2 Yoneda Projective

Consider the concrete category



and its $\mathbb{K}$-Algebroid kq



together with the relations

$$[a^4-(1), e^2-(2), h^2-(3), j^1-(4), bf-c, bef-ach, bg-d, ci-d, achi-beg, a^3beg-chi, fi-g]$$

The resulting category algebra has dimension 43.

We can look at the submodule of the category algebra consisting of all arrows starting at `kq.1`. This is what the function `YonedaProjective( CatReps, kq.1 )` gives us:

```
proj1 := YonedaProjective( CatReps, kq.1 ); <(1)->4, (2)->8, (3)->8, (4)->8; (a)->4x4,
(b)->4x8, (c)->4x8, (d)->4x8, (e)->8x8, (f)->8x8, (g)->8x8, (h)->8x8, (i)->8x8, (4)->8x8>
```

The number 4 associated with object (1) tells us that the submodule of all arrows starting and ending at (1) has dimension 4. Its basis is the set of paths $\{a, a^2, a^3, a^4 = (1)\}$.

Likewise in

```
proj4 := YonedaProjective( CatReps, kq.4 ); <(1)->0, (2)->0, (3)->0, (4)->1; (a)->0x0,
(b)->0x0, (c)->0x0, (d)->0x1, (e)->0x0, (f)->0x0, (g)->0x1, (h)->0x0, (i)->0x1, (4)->1x1>
```

The submodule of all arrows starting at (4) is only of dimension 1, since it's already the identity arrow $\{j = (4)\}$.

Dimension of the (quotient of the) path algebra is 43. Sum of all dimensions of the yoneda projectives on each objects is 43.

Definition 7.2.1. (Yoneda projective) Yoneda's projective representation given by the object o is the submodule of the category algebra consisting of all arrows starting at o.

Conjecture:
Dimension of the path algebra = Sum of dimensions of the yoneda projectives on each object.
What does the yoneda projective mean???
Function that creates examples for concrete categories so that I can check my conjecture.

8 Functors and natural transformations

8.1 Functors act on objects and morphisms of a category

8.2 Natural transformations are morphisms between functors

8.3 Representations are Functors into a matrix category

8.4 Finite concrete categories

9 Algorithms

Algorithm 2: `ConvertToMapOfFinSets`

Input : a list *objects* of objects in FinSets and a morphism *gen* given as a list of images in the convention of catreps
Output : the corresponding map of finite sets from source S to target T

let T be the first object $O \in objects$ such that $gen \cap O \neq \varnothing$;
if $gen \cap O = \varnothing \ \forall O \in objects$ **then**
  Error "unable to find target set"
end
let fl be the flattening of *objects* as a list;
let S be the sublist of fl according to positions i such that $gen[i]$ is bound;
set S to be the first object $O \in objects$ such that $O = S$;
if $S \neq O \ \forall O \in objects$ **then**
  Error "unable to find source set"
end
let G be the list of pairs $[i, gen[i]], i \in S$;
return MapOfFinSets(S, G, T);

We can now create finite concrete categories with objects not starting from 1, to demonstrate that `ConcreteCategoryForCAP( [ [,,,5,6,4], [,,,7,8,9], [,,,,,,,8,9,7] ] )` and `ConcreteCategoryForCAP( [ [2,3,1], [4,5,6], [,,,5,6,4] ] )` yield equivalent categories, i.e. their underlying quivers are the same and they give the same category of representations.

Yonedas Einbettungs-Lemma: Fehlende Limiten bzw. Kolimiten exitieren nach der Einbettung.

Einbettung in Kategorien, die mehr Limiten haben als die Zielkategorie.

"(Ko-)Vervollständigung" der Kategorie (Completion / Cocompletion)

Quiver = unvollständige Struktur einer Kategorie Erzeugendensystem einer Kategorie.

K-linearer Abschluss einer Kategorie

Pfadalgebra = Kategorien-Algebra path algebra = 1 Object, welches eine Algebra ist. Dabei verliert man wieder die Informationen über die mehreren Objekte.

So wie Menge ein Erz-system eines Monoid.

Algorithm 3: `RightQuiverFromConcreteCategory`

Input : a finite concrete category C with n objects
Output : the right quiver $q(n)$

let Obj be the set of objects of C;
let $n := Length(Obj)$;
let $gMor$ be the set of generating morphisms of C;
let A be the empty set and let $i := 1$;
foreach *morphism mor in gMor* **do**
  let $A_{i,1}$ be the position of $Source(mor)$ in Obj;
  let $A_{i,2}$ be the position of $Range(mor)$ in Obj;
  let $i := i + 1$;
end
let q be the right quiver with vertices $\{1, \ldots, n\}$ and arrows A.
return q;

Algorithm 4: `RelationsOfEndomorphisms`

Input: a commutative ring k and a finite concrete category C
Output: the endomorphism relations of the category C

let $q :=$ `RightQuiverFromConcreteCategory`(C);
let kq be the path algebra generated by k and q;
let $gMor$ be the set of generating morphisms of C;
let $A := Arrows(q)$;
let $relsEndo$ be the empty set;
foreach $i = 1, \ldots, Length(gMor)$ **do**
  let $mor := gMor_i$
  if *mor is not an endomorphism* **then**
    continue;
  end
  let $m := 0$ and let $powers$ be the empty set;
  let $foundEqual$ be false;
  while $mor^m \notin powers$ **do**
    let $n := 1$;
    while $\neg foundEqual$ **do**
      if $mor^{(m+n)} = mor^m$ **then**
        Add the relation $kq.(A_i)^{(m+n)} - kq.(A_i)^m$ to relsEndo;
        foundEqual := true;
      end
      n := n+1;
    end
    Add mor^m to powers;
    m := m+1;
  end
end
return $relsEndo$;

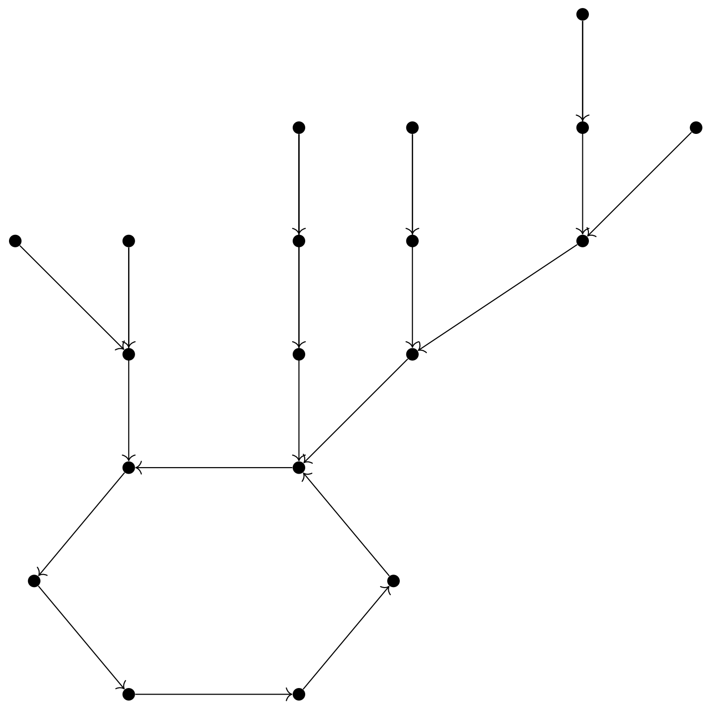
Algorithm 5: `Algebroid`

Input: a commutative ring k and a finite concrete category C
Output: the k-linear closure of the category C over the commutative ring k

return;

10 Relations of the Algebroid

10.1 Relations of endomorphisms



Lemma 10.1.1 (σ-Lemma). *Let $\mathcal{C}$ be a finite concrete category. Then for each object $M \in \mathcal{C}_0$ the set $\mathrm{End}_{\mathcal{C}}(M)$ is a monoid and for each endomorphism $f \in \mathrm{End}_{\mathcal{C}}(M)$ there exist $m, n \in \mathbb{N}$ such that $f^{(m+n)} = f^m$. If $m = 0$ and $n \geq 1$ then f is bijective with $f^{-1} = f^{n-1}$.*

Proof. The properties of a monoid are precisely the associativity of composition and the unit property from 3 and 4. Since $|\mathrm{End}_{\mathcal{C}}(M)| < \infty$ there are only finitely many endomorphisms $f_1, \ldots, f_N \in \mathrm{End}_{\mathcal{C}}(M)$. Let $\{f^k | k \in \mathbb{N}\} \subset \mathrm{End}_{\mathcal{C}}(M)$, i.e. there is a function $\{f^k | k \in \mathbb{N}\} \to \{f_j | j \in \{1, \ldots, N\}\}; f^k \mapsto f_j$ not necessarily surjective and by the pigeonhole principle highly non injective, since $|\mathbb{N}| > |\mathrm{End}_{\mathcal{C}}(M)|$. Let $m := Min\{k \in \mathbb{N} | f^k = f_j\}$

□

Lemma 10.1.2. *Algorithm 8 terminates and yields the correct result.*

Proof. Since $\mathcal{C}$ is a finite concrete category, for each object $M \in \mathcal{C}_0$ the endomorphism set $\mathrm{End}_{\mathcal{C}}(M)$ is finite. If in step i, $f := gMor_i \in \mathrm{End}_{\mathcal{C}}(M)$ is an endomorphism, then $\{f, f^2, f^3, \ldots\}$ is a subset of the finite set $\mathrm{End}_{\mathcal{C}}(M)$, therefore $\exists N \in \mathbb{N}$ such that $f^k \in \{f, f^2, f^3, \ldots, f^N\}\ \forall k \geq N$. This proves that the sets *mpowers* and *npowers*, which contain increasing powers of f, will be finite and thus at some point already contain f^m or f^{m+n} respectively, causing the while loops to terminate. The if clause makes sure that the set *relsEndo* only contains the desired relations. It remains to be shown that those are all the endomorphism relations. □

Beschreibung der Algorithmen
WeakDirectSumDecomposition ¡– Tiefensuche. Objekte (Funktoren) in indecomposable Functors.

Example 10.1.3. (Representation of a concrete category)

Algorithm 6: `RelationsOfEndomorphisms`

Input: a commutative ring k and a finite concrete category $\mathcal{C}$
Output: the endomorphism relations of the category $\mathcal{C}$

let q := `RightQuiverFromConcreteCategory`$(\mathcal{C})$;
let kq be the path algebra generated by k and q;
let $gMor$ be the set of generating morphisms of $\mathcal{C}$;
let $A := Arrows(q)$;
set $relsEndo := \varnothing$;
foreach $i = 1, \ldots, Length(gMor)$ **do**
  let $f := gMor_i$
  if f *is not an endomorphism* **then**
    continue;
  end
  let $m := 0$;
  set $mpowers := \varnothing$;
  let $foundEqual$ be false;
  while $f^m \notin mpowers$ **do**
    let $n := 1$;
    set $npowers := \varnothing$;
    while $\neg foundEqual$ *and* $f^{m+n} \notin npowers$ **do**
      if $f^{m+n} = f^m$ **then**
        Add the relation $kq.(A_i)^{m+n} - kq.(A_i)^m$ to relsEndo;
        foundEqual := true;
      end
      Add f^{m+n} to npowers;
      n := n+1;
    end
    Add f^m to mpowers;
    m := m+1;
  end
end
return *relsEndo*;

$$\begin{tikzcd}
5 \arrow[r, "\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}"] \arrow[loop below, "\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}"] & 4 \arrow[loop below, "\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}"]
\end{tikzcd}$$

$$nine \Uparrow$$

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

$$\begin{tikzcd}
\{1,2,3\} \arrow[r, "(4{,}5{,}6)"] \arrow[loop below, "(2{,}1{,}3)"] & \{4,5,6\} \arrow[loop below, "(5{,}6{,}4)"]
\end{tikzcd}$$

$$F(a)\eta_1 = \eta_1 G(a) F(b)\eta_2 = \eta_1 G(b)$$

11 $\mathbb{K}$-linear Category (Algebroid)

Group: Category with one object.

Groupoid: A small category in which every morphism is an isomorphism.

Algebroid

EmbeddingOfSumOfImages

What is an Algebroid? Bialgebroid?

12 Additive Category

13 Abelian Category

14 The Category of Categories

15 The Categories of Functors

16 The Representation of a Category

17 Representation

Grundidee von FunctorCategory
  Standard-Monoidale Struktur von der Zielkategorie z.B. TensorUnit(C)

18 Algorithms

```
    end );

    AddInverse( C,
      function( alpha )
        return Inverse( UnderlyingCell( alpha ) ) / CapCategory( alpha );
    end );

    c := ConcreteCategory( L );

    C!.ConcreteCategoryRecord := c;

    objects := List( c.objects, FinSet );

    SetSetOfObjects( C, List( objects, o -> o / C ) );

    SetSetOfGeneratingMorphisms( C, List( c.generators, g -> ConvertToMapOfFinSets( obje

    Finalize( C );

    return C;

end );

##
InstallMethod( Algebroid,
        "for a homalg ring and a finite category",
        [ IsHomalgRing and IsCommutative, IsFiniteConcreteCategory ],

  function( k, C )
    local objects, gmorphisms, q, kq, relEndo, A, F, vertices, rel,
          func, st, s, t, homST, list, p, pos;

    objects := SetOfObjects( C );
    gmorphisms := SetOfGeneratingMorphisms( C );
    q := RightQuiverFromConcreteCategory( C );
    kq := PathAlgebra( k, q );
    relEndo := RelationsOfEndomorphisms( k, C );
    A := Algebroid( kq, relEndo );
```

```
    kq := UnderlyingQuiverAlgebra( A );
    F := CapFunctor( A, objects, gmorphisms, C );

    vertices := List( SetOfObjects(A), UnderlyingVertex );

    rel := [];
    func :=
      function( p, l )
        return ForAny( l, p1->
                       IsCongruentForMorphisms(
                               ApplyToQuiverAlgebraElement( F, p ),
                               ApplyToQuiverAlgebraElement( F, p1 ) )
                       );
    end;

    for st in Cartesian(vertices,vertices) do
        s := st[1];
        t := st[2];
        if s = t then
            continue;
        fi;
        homST := BasisPathsBetweenVertices( kq, s, t );
        homST := List( homST, p -> PathAsAlgebraElement( kq, p ) );

        list := [];

        for p in homST do
            pos := PositionProperty( list, l->func(p,l) );
            if IsInt(pos) then
                Add( list[pos], p );
            else
                Add( list, [p] );
            fi;
        od;
        list := List( list, l-> List( l, p -> p!.representative ) );
        Append( rel, list );
    od;

    rel := Filtered( rel, l -> Length(l)>1 );
    rel := List( rel, l -> List( l{[ 2 .. Length(l) ]}, p -> l[1]-p ) );
    rel := Flat( rel );
    rel := Concatenation( relEndo, rel );

    kq := PathAlgebra( kq ) / rel;
```

We want the endomorphism relations so that the path algebra is finite-dimensional and we get a finite Gröbner basis.

Proof that algorithm is correct Proof that it terminates.

Wir haben BasisOfExternalHom benutzt um Decompose in CAP umzusetzen um EmbeddingOfSubRepresentation umzusetzen um WeakDirectSumDecomposition umzusetzen.

Notes

Algorithm 7: `RightQuiverFromConcreteCategory`

Input : a finite concrete category C with n objects
Output : the right quiver $q(n)$

let Obj be the set of objects of C;
let $n := Length(Obj)$;
let $gMor$ be the set of generating morphisms of C;
let A be the empty set and let $i := 1$;
foreach *morphism mor in gMor* **do**
  let $A_{i,1}$ be the position of $Source(mor)$ in Obj;
  let $A_{i,2}$ be the position of $Range(mor)$ in Obj;
  let $i := i + 1$;
end
let q be the right quiver with vertices $\{1, \dots, n\}$ and arrows A.
return q;

Algorithm 8: `RelationsOfEndomorphisms`

Input: a commutative ring k and a finite concrete category C
Output: the endomorphism relations of the category C

let $q :=$ `RightQuiverFromConcreteCategory`(C);
let kq be the path algebra generated by k and q;
let $gMor$ be the set of generating morphisms of C;
let $A := Arrows(q)$;
let $relsEndo$ be the empty set;
foreach $i = 1, \dots, Length(gMor)$ **do**
  let $mor := gMor_i$ **if** *mor is not an endomorphism* **then**
    continue;
  end
  let $m := 0$ and let $powers$ be the empty set;
  let $foundEqual$ be false;
  while $mor^m \notin powers$ **do**
    let $n := 1$;
    while $\neg foundEqual$ **do**
      if $mor^{(m+n)} = mor^m$ **then**
        Add the relation $kq.(A_i)^{(m+n)} - kq.(A_i)^m$ to relsEndo;
        foundEqual := true;
      end
      n := n+1;
    end
    Add mor^m to powers;
    m := m+1;
  end
end
return *relsEndo*;

References

[1] https://web.northeastern.edu/martsinkovsky/p/Parnu2019/slides-facchini.pdf

[2] https://www.math.uni-bielefeld.de/ sek/kau/leit4.pdf

[3] Jan Geuenich. https://hss.ulb.uni-bonn.de/2017/4681/4681.pdf

[4] Mohamed Barakat, Julia Mickisch and Fabian Zickgraf, `FinSetsForCAP`, https://github.com/mohamed-barakat/FinSetsForCAP/ (Retrieved: 1 April 2020)