Bachelorarbeit Representations of Concrete Categories as Objects in the Functor Category

Tibor Grün

June 16, 2020

Contents

1 Einleitung

Wie viele Vokablen braucht man, um Mathematik zu betreiben?

$$\mathbb{K}^{\{1,2,3\}} \leq \mathbb{K}^{\{1,2,3,4\}}$$

$$\{1,2,3\} \subset \{1,2,3,4\}$$

$$3 < 4$$

Diese drei Sachverhalte drücken eigentlich alle dasselbe aus.

Denken in Kategorien, aber kleinteilig auf Elementebene programmieren? Wenn wir die Kontrolle über unsere Datenstrukturen nicht abgeben wollen, sondern versuchen, im Quellcode selbst zu optimieren geht das unweigerlich aufkosten der Lesbarkeit des Codes. Und wenn die kleinen Performance-Gewinne gegenüber dem Platzhirsch in ein paar Jahren dadurch aufgehoben werden, weil sich andere Experten allein um Optimierung gekümmert haben, stellt man fest, man hat aufs falsche Pferd gesetzt. Denken in Kategorien und programmieren in Kategorien. Man geht ein, zwei, beliebig viele Abstraktionsstufen höher, und überlässt die eigentliche Rechnung dafür optimierten Programmen. Wenn ich einen Kern einer Matrix berechne, die Matrix selbst das Ergebnis komplizierter kategorientheoretischer Konstruktionen ist, dann ist es wichtiger zu wissen, was die Matrix ist, wie sie entstanden ist, als das Ergebnis der Berechnungen elementweise zu kennen.

Mit dem `CAP`-Project haben [Posur et al.] gezeigt, dass es möglich ist, kategoriell zu denken und zu programmieren, und dabei die eigentliche Rechenarbeit an optimierte Computeralgebraprogramme wie Singular oder Magma auszugliedern. Man muss bei der Rechnung eben nicht von Grund auf anfangen. Das bringt uns zurück zur Frage, wie viele Vokablen braucht man, um Mathematik zu betreiben?

Die Mengentheorie kommt aus mit dem Element-Symbol $\in$ und den Mengenklammern $\{$ und $\}$, sowie mit den Prädikaten der Logik. Nimmt man dann noch ein paar Axiome hinzu, kommt man zu Axiomensystemen wie Zermelo-Fraenkel (ZF) oder ZFC.

$$\mathbf{Quiv} \rightarrow^{CatClosure} \leftarrow_U \mathbf{Cats} \rightarrow^{k-Algebroid} \leftarrow_U \mathbf{k-Cats} \rightarrow^{AdditiveClosure} \leftarrow_U \mathbf{k-Cats}^{\oplus}$$

2 Directed Quiver, Path Algebra and the Algebroid

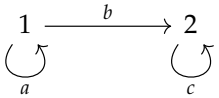
Definition 2.1. (Quiver)
A directed quiver q consists of a class of objects (or vertices) $q_0 = \operatorname{Obj} q$ and a class of morphisms (or arrows) $q_1 = \operatorname{Mor} q$ together with two defining maps

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

s called source and t called target.

In `QPA`, the objects are coded by natural numbers, so the first object is 1, the second 2 and so on. The arrows are denoted by small letters a, b, c and so on. There is a difference between `RightQuiver` and `LeftQuiver` in that the right quiver is right-oriented (that is, the convention for order in multiplication of paths is the opposite of that used for left-oriented quivers).

Example 2.2. (Quiver with 2 objects and 3 morphisms)



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 $s(b) = 1, t(b) = 2$.

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.

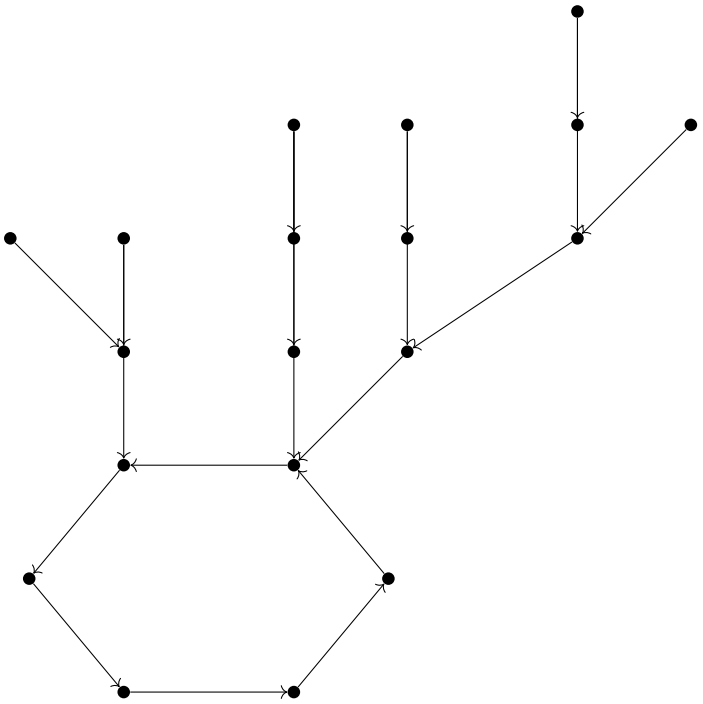
Lemma 2.3. *Let Q be a quiver. If there is a path of length at least $|Q_0|$, then there are cyclic paths, and thus infinitely many paths.*

Proof. Assume that there exists a path of length greater or equal to $|Q_0|$. Then there exists a path of length $\geq |Q_0|$, say $alpha_n \cdots alpha_1$. 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 $\omega = alpha_{j-1} \cdots alpha_i$, this is a path with target and source $x_i = x_j$, 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. □

Path Algebra:
(unvollständige Struktur einer Kategorie) Erzeugendensystem einer Kategorie.
K-linearer Abschluss einer Kategorie
Pfadalgebra = Kategorien-Algebra
So wie Menge ein Erz-system eines Monoid.

3 Relations of the Algebroid

3.1 Relations of endomorphisms



Lemma 3.1 (σ-Lemma)**.** *For each endomorphism f in a finite concrete category $\mathcal{C}$ there exist $m, n \in \mathbb{N}$ such that $f^{(m+n)} = f^m$.*

4 Category

Definition 4.1. (Quiver)
A quiver A consists of a class of objects (or vertices) $A_0 = \mathrm{Obj}A$ and a class of morphisms (or arrows) $A_1 = \mathrm{Mor}A$ together with two defining maps

$$s, t \colon A_1 \rightrightarrows A_0$$

s called source and t called target.[1]
We write $\mathrm{Hom}_A(M, N)$ (sometimes also $A(M, N)$) for the fiber $(s,t)^{-1}(\{(M,N)\})$ of the product map $(s,t) : A_1 \to A_0 \times A_0$ over the pair $(M, N) \in A_0 \times A_0$.
This is the class of all morphisms with source = M and target = N.
For a morphism $\varphi \in \mathrm{Hom}_A(M, N)$ we write

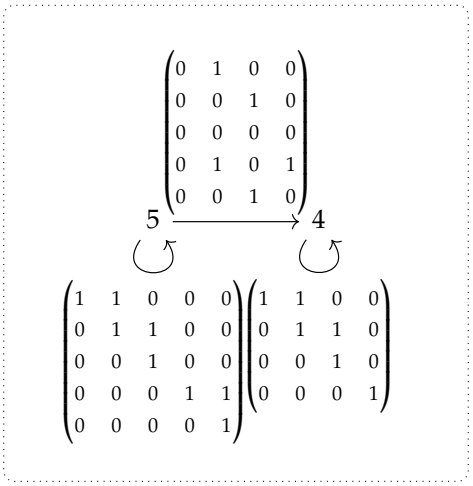
$$\varphi : M \longrightarrow N \text{ or } M \xrightarrow{\varphi} N$$

Clearly A_1 is the disjoint union $\dot{\bigcup}_{M,N\in A_0} \mathrm{Hom}_A(M,N) = A_1$. As usual we define $\mathrm{End}_A(M) := \mathrm{Hom}_A(M,M)$.

Definition 4.2. (Category)
A <u>category</u> $\mathfrak{A}$ is a quiver with two further defining maps

$$A_0 \xrightarrow{1} A_1 \xleftarrow[\mu]{} A_1 \times_{s,A_0,t} A_1$$

Example 4.3. (Representation of a concrete category)



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

5 $\mathbb{K}$-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?

6 Additive Category

7 Abelian Category

8 The Category of Categories

9 The Categories of Functors

10 The Representation of a Category

11 Representation

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

12 Algorithms

```
60      AddInverse( C,
61        function( alpha )
62          return Inverse( UnderlyingCell( alpha ) ) / CapCategory( alpha );
63      end );
64
65      c := ConcreteCategory( L );
66
67      C!.ConcreteCategoryRecord := c;
68
69      objects := List( c.objects, FinSet );
70
71      SetSetOfObjects( C, List( objects, o -> o / C ) );
72
73      SetSetOfGeneratingMorphisms( C, List( c.generators, g -> ConvertToMapOfFinSets( obje
74
75      Finalize( C );
76
77      return C;
78
79  end );
80
81  ##
82  InstallMethod( Algebroid,
83          "for a homalg ring and a finite category",
84          [ IsHomalgRing and IsCommutative, IsFiniteConcreteCategory ],
85
86    function( k, C )
87      local objects, gmorphisms, q, kq, relEndo, A, F, vertices, rel,
88            func, st, s, t, homST, list, p, pos;
89
90      objects := SetOfObjects( C );
91      gmorphisms := SetOfGeneratingMorphisms( C );
92      q := RightQuiverFromConcreteCategory( C );
93      kq := PathAlgebra( k, q );
94      relEndo := RelationsOfEndomorphisms( k, C );
95      A := Algebroid( kq, relEndo );
96      kq := UnderlyingQuiverAlgebra( A );
97      F := CapFunctor( A, objects, gmorphisms, C );
```

```
98
99      vertices := List( SetOfObjects(A), UnderlyingVertex );
100
101     rel := [];
102     func :=
103       function( p, l )
104         return ForAny( l, p1->
105                        IsCongruentForMorphisms(
106                                ApplyToQuiverAlgebraElement( F, p ),
107                                ApplyToQuiverAlgebraElement( F, p1 ) )
108                        );
109     end;
110
111     for st in Cartesian(vertices,vertices) do
112         s := st[1];
113         t := st[2];
114         if s = t then
115             continue;
116         fi;
117         homST := BasisPathsBetweenVertices( kq, s, t );
118         homST := List( homST, p -> PathAsAlgebraElement( kq, p ) );
119
120         list := [];
121
122         for p in homST do
123             pos := PositionProperty( list, l->func(p,l) );
124             if IsInt(pos) then
125                 Add( list[pos], p );
126             else
127                 Add( list, [p] );
128             fi;
129         od;
130         list := List( list, l-> List( l, p -> p!.representative ) );
131         Append( rel, list );
132     od;
133
134     rel := Filtered( rel, l -> Length(l)>1 );
135     rel := List( rel, l -> List( l{[ 2 .. Length(l) ]}, p -> l[1]-p ) );
136     rel := Flat( rel );
137     rel := Concatenation( relEndo, rel );
138
139     kq := PathAlgebra( kq ) / rel;
140
141     kq := PathAlgebra( kq ) / GroebnerBasis( IdealOfQuotient( kq ) );
```

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

[1]Some authors use maps t, h for *tail* and *head* instead of source and target, defining the arrows to go from the tail to the head. This use of t as the starting point instead of the end target as in our definition can lead to some confusion.

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;

Algorithm 2: `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*;

References

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