A Coq Formalization of Finitely Presented Modules

Cyril Cohen and Anders Mörtberg

Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
{cyril.cohen,anders.mortberg}@cse.gu.se

Abstract. This paper presents a formalization of constructive module theory in the intuitionistic type theory of Coq. We build an abstraction layer on top of matrix encodings, in order to represent finitely presented modules, and obtain clean definitions with short proofs justifying that it forms an abelian category. The goal is to use it as a first step to compute certified topological invariants, like homology groups and Betti numbers.

Keywords: Formalization of mathematics, Homological algebra, Constructive algebra, Coq, SSReflect

1 Introduction

Homological algebra is the study of linear algebra over rings instead of fields, this means that one consider modules instead of vector spaces. Homological techniques are ubiquitous in many branches of mathematics like algebraic topology, algebraic geometry and number theory. Homology was originally introduced by Henri Poincaré in order to compute topological invariants of spaces [23], which provides means for testing whether two spaces cannot be continuously deformed into one another. This paper presents a formalization[1] of constructive module theory in type theory, using the Coq proof assistant [7] together with the Small Scale Reflection (SSReflect) extension [13], which provides a potential core of a library of certified homological algebra.

In a previous work one of the authors explored ways to compute homology groups of vector spaces [17,18] in Coq. When generalizing this to commutative rings the universal coefficient theorem of homology [15] states that most of the homological information of a module over a ring R can be computed by only doing computation with elements in $\mathbb{Z}$. This means that if we were only be interested in computing homology it would not really be necessary to develop the theory of modules in general but instead do it for $\mathbb{Z}$-modules which are well behaved because any matrix can be put in Smith normal form. However, by developing the theory for general rings it should be possible to implement and reason about

[1] The formal development is at: http://perso.crans.org/cohen/work/fpmods/

other functors like cohomology, Ext and Tor as in the HOMALG computer algebra package [3].

In [12], Georges Gonthier shows that the theory of finite dimensional vector spaces can be elegantly implemented in COQ by using matrices to represent subspaces and morphisms, as opposed to an axiomatic approach. The reason why abstract finite dimensional linear algebra can be concretely represented by matrices is because any vector space has a basis (a finite set of generators with no relations among the generators) and any morphism can be represented by a matrix in this canonical basis. However, for modules over rings this is no longer true[2], this means that the matrix-based approach cannot be directly applied when formalizing module theory. This is why we restrict our attention to finitely generated modules that are *finitely presented*, that is, modules with a finite number of generators and a finite number of relations among these generators. In constructive module theory one usually restricts attention to this kind of modules and all algorithms can be described by manipulating the presentation matrices [10,14,20,22]. This paper can hence be seen as a generalization of the formalization of Gonthier to modules over rings instead of fields.

At the heart of the formalization of Gonthier is an implementation of Gaussian elimination which is used in all subspace constructions. In particular from it we can compute the kernel which characterizes the space of solutions of a system of linear equations. However when doing module theory over arbitrary rings, there is no general algorithm for solving systems of linear equations. This is why we restrict our attention to modules over *coherent* and *strongly discrete rings*, as is customary in constructive algebra [20,22], which means that we can solve systems of equations.

The main contributions of this paper are the representation of finitely presented modules over coherent strongly discrete rings (Sect. 2), basic operations on these modules (Sect. 3) and we show that the collection of these modules and morphisms forms an abelian category (Sect. 4), which means that it is a suitable setting for developing homological algebra. We have also proved that, over Bézout domains where every matrix has a Smith normal form, it is possible to test if two finitely presented modules represent isomorphic modules (Sect. 5). (Examples of such Bézout domains are principal ideal domains like $\mathbb{Z}$ and $k[x]$ where k is a field).

2 Finitely presented modules

As mentioned in the introduction, a module is finitely presented if it can be given by a finite set of generators and relations. This is traditionally expressed as:

Definition 1. *A R-module $\mathcal{M}$ is **finitely presented** if there is an exact sequence:*

[2] Consider the ideal (X, Y) of $k[X, Y]$, it is a module generated by X and Y which is not free because $XY = YX$.

$$R^{m_1} \xrightarrow{M} R^{m_0} \xrightarrow{\pi} \mathcal{M} \longrightarrow 0$$

More precisely, π is a surjection and M a matrix representing the m_1 relations among the m_0 generators of the module $\mathcal{M}$. This means that $\mathcal{M}$ is the cokernel of M:

$$\mathcal{M} \simeq \mathrm{coker}(M) = R^{m_0}/im(M)$$

Hence a module has a finite presentation if it can be expressed as the cokernel of a matrix. As all information about a finitely presented module is contained in its presentation matrix we will omit the surjection π when giving presentations of modules.

Example 1. The $\mathbb{Z}$-module $\mathbb{Z} \oplus \mathbb{Z}/2\mathbb{Z}$ is given by the presentation:

$$\mathbb{Z} \xrightarrow{\begin{pmatrix} 0 & 2 \end{pmatrix}} \mathbb{Z}^2 \longrightarrow \mathbb{Z} \oplus \mathbb{Z}/2\mathbb{Z} \longrightarrow 0$$

as if $\mathbb{Z} \oplus \mathbb{Z}/2\mathbb{Z}$ is generated by (e_1, e_2) there is one relation, namely $0e_1 + 2e_2 = 2e_2 = 0$.

Operations on finitely presented modules can now be implemented by manipulating the presentation matrices, for instance if $\mathcal{M}$ and $\mathcal{N}$ are finitely presented R-modules:

$$R^{m_1} \xrightarrow{M} R^{m_0} \longrightarrow \mathcal{M} \longrightarrow 0 \qquad R^{n_1} \xrightarrow{N} R^{n_0} \longrightarrow \mathcal{N} \longrightarrow 0$$

the presentation of $\mathcal{M} \oplus \mathcal{N}$ is given by:

$$R^{m_1+n_1} \xrightarrow{\begin{pmatrix} M & 0 \\ 0 & N \end{pmatrix}} R^{m_0+n_0} \longrightarrow \mathcal{M} \oplus \mathcal{N} \longrightarrow 0$$

We have represented finitely presented modules in Coq using the datastructure of matrices from the SSReflect library which is defined as:

```
Inductive matrix R m n := Matrix of {ffun 'I_m * 'I_n -> R}.

(* With notations: *)
(* 'M[R]_(m,n) = matrix R m n *)
(* 'rV[R]_m = 'M[R]_(1,m) *)
(* 'cV[R]_m = 'M[R]_(m,1) *)
```

where `'I_m` is the type `ordinal m` which represents all natural numbers smaller than `m`. This type has exactly `m` inhabitants and can be coerced to the type of natural numbers, `nat`. Matrices are then represented as finite functions over the finite set of indices, which means that dependent types are used to express well-formedness. Finitely presented modules are now conveniently represented using a record containing a matrix and its dimension:

```
Record fpmodule := FPModule {
  nbrel : nat;
  nbgen : nat;
  pres : 'M[R]_(nbrel, nbgen)
}.
```

The direct sum of two finitely presented modules is now straightforward to implement:

```
Definition dsum (M N : fpmodule R) :=
  FPModule (block_mx (pres M) 0 0 (pres N)).
```

Here `block_mx` forms the block matrix consisting of the four submatrices. We now turn our attention to morphisms of finitely presented modules.

2.1 Morphisms

As for vector spaces we represent morphisms of finitely presented modules using matrices. The following lemma states how this can be done:

Lemma 1. *If $\mathcal{M}$ and $\mathcal{N}$ are finitely presented R-modules given by presentations:*

$$R^{m_1} \xrightarrow{M} R^{m_0} \longrightarrow \mathcal{M} \longrightarrow 0 \qquad R^{n_1} \xrightarrow{N} R^{n_0} \longrightarrow \mathcal{N} \longrightarrow 0$$

and $\varphi : M \to N$ a module morphism then there is a $m_0 \times n_0$ matrix φ_G and a $m_1 \times n_1$ matrix φ_R such that the following diagram commutes:

$$\begin{array}{ccccccc} R^{m_1} & \xrightarrow{M} & R^{m_0} & \longrightarrow & \mathcal{M} & \longrightarrow & 0 \\ \downarrow{\scriptstyle \varphi_R} & & \downarrow{\scriptstyle \varphi_G} & & \downarrow{\scriptstyle \varphi} & & \\ R^{n_1} & \xrightarrow{N} & R^{n_0} & \longrightarrow & \mathcal{N} & \longrightarrow & 0 \end{array}$$

For a proof of this see Lemma 2.1.25 in [14]. This means that morphisms between finitely presented modules can be represented by pairs of matrices. The intuition why two matrices are needed is that the morphism affects both the generators and relations of the modules, hence the names φ_G and φ_R.

In order to be able to compute for example the kernel of a morphism of finitely presented modules we need to add some constraints on the ring R since, in general, there is no algorithm for solving systems of equations. The class of rings that we want to consider are *coherent* and *strongly discrete* which means that it is possible to solve systems of equations, in HOMALG these are called *computable rings* [2] and form the basis of the system.

2.2 Coherent and strongly discrete rings

Given a ring R (in our setting commutative but it is possible to consider non-commutative rings as well [2]) we want to study the problem of

solving linear systems over R. If R is a field we have a nice description of the space of solution by a basis of solutions. Over an arbitrary ring R there is in general no basis.[3] An important weaker property is that there is a finite number of solutions which generate all solutions.

Definition 2. *A ring is **(left) coherent** if for any matrix M it is possible to compute a matrix L such that:*

$$XM = 0 \;\leftrightarrow\; \exists Y. X = YL$$

This means that L generates the module of solutions of $XM = 0$, hence L is the kernel of M. For this it is enough to consider the case where M has only one column [20]. Note that the notion of coherent rings is not stressed in classical presentations of algebra since Noetherian rings are automatically coherent, but in a computationally meaningless way. It is however a fundamental notion, both conceptually [20,22] and computationally [3].

One of the authors previously represented coherent rings in Coq [8], however there are some minor differences in this implementation in order to make it convenient to work with finitely presented modules. One difference is that in the previous presentation only *right* coherent rings (the indeterminate were on the right, i.e. $MX = 0$) were represented while here we have *left* coherent rings. The reason is that SSReflect takes the convention that composition of linear applications is done in diagrammatic order (i.e. in the order they appear in the diagram).

In the development, coherent rings have been implemented as in [11] using the `Canonical Structure` mechanism of Coq. As matrices are represented using dependent types denoting their size this need to be known when defining coherent rings. In general the size of L cannot be predicted, so we include an extra function to compute this:

```
Record mixin_of (R : ringType) : Type := Mixin {
  dim_ker : forall m n, 'M[R]_(m,n) -> nat;
  ker : forall m n (M : 'M_(m,n)), 'M_(dim_ker M,m);
  _ : forall m n (M : 'M_(m,n)) (X : 'rV_m),
      reflect (exists Y, X = Y *m ker M) (X *m M == 0)
}.
```

Here `*m` denotes matrix multiplication and `==` is the boolean equality of matrices, so the specification says that this equality is equivalence to the existence statement. An alternative to having a function computing the size would be to output a dependent pair but this has the undesirable behavior that the pair has to be destructed when stating lemmas about it, which in turn would make these lemmas cumbersome to use as it would not be possible to rewrite with them directly.

An algorithm that can be implemented using `ker` is the kernel modulo a set of relation, that is, computing $\ker(R^m \xrightarrow{M} \mathrm{coker}(N))$. This is

[3] For instance over the ring $R = k[X, Y, Z]$ where k is a field, the equation $pX + qY + rZ = 0$ has no basis of solutions. It can be shown that a generating system of solutions is given by $(-Y, X, 0),\ (Z, 0, -X),\ (0, -Z, Y)$.

equivalent to finding a X such that $\exists Y, XM+YN = 0$, which is the same as solving $(X\ Y)(M\ N)^T = 0$ and returning the part of the solution that corresponds to XM. We denote this as $\ker_N(M)$ and as `N.-ker(M)` in the formalization. Note that this is a more fundamental operation than taking the kernel of a matrix in the since $XM = 0$ is also equivalent to $\exists Y, X = Y \ker_0(M)$

In order to test if a module is zero we also need to be able to solve systems of the kind $XM = B$ where B is not zero. In order to do this we need to introduce another kind of rings that are important in constructive algebra:

Definition 3. *A ring R is **strongly discrete** if membership in finitely generated ideals is decidable and if $x \in (a_1, \ldots, a_n)$ there is an algorithm computing $w_1, \ldots, w_n$ such that $x = \sum_i a_i w_i$.*

Examples of such rings are multivariate polynomial rings over fields with decidable equality (via Gröbner bases) and Bézout domains (for instance $\mathbb{Z}$ and $k[x]$). This kind of rings has been implemented in a similar way as coherent rings.

If a ring is both coherent and strongly discrete it is not only possible to solve homogeneous systems $XM = 0$ but also any system $XM = B$ where B is an arbitrary matrix with the same number of columns as M. This operation can be seen as division of matrices as:

```
Lemma dvdmxP m n k (M : 'M[R]_(n,k)) (B : 'M[R]_(m,k)) :
  reflect (exists X, X *m M = B) (M %| B).
```

Here `%|` is notation for the function computing the particular solution to $XM = B$, returning `None` in the case that a solution does not exist. We have developed a library of divisibility of matrices with lemmas like

```
Lemma dvdmxD m n k (M : 'M[R]_(m,n)) (N K : 'M[R]_(k,n)) :
  M %| N -> M %| K -> M %| N + K.
```

which follow directly from `dvdmxP`. This can now be used to give a nicer representation of morphisms of finitely presented modules and the division theory of matrices gives short and elegant proofs about operations on morphisms.

2.3 Finitely presented modules over coherent strongly discrete rings

As mentioned before, morphisms between finitely presented R-modules $\mathcal{M}$ and $\mathcal{N}$ can be represented by a pair of matrices. However when R is coherent and strongly discrete it suffices to only consider the φ_G matrix as φ_R can be computed by solving $XN = M\varphi_G$, which is the same as testing $N \mid M\varphi_G$. In Coq this means that morphisms between two finitely presented modules can be implemented as:

```
Record morphism_of (M N : fpmodule R) := Morphism {
  matrix_of_morphism : 'M[R]_(nbgen M,nbgen N);
  _ : pres N %| pres M *m matrix_of_morphism
```

```
}.

(* With notation: *)
(* 'Mor(M,N) := morphism_of M N *)
```

Using this representation we can easily define the identity morphism (`idm`) and composition of morphisms (`phi ** psi`) and show that these form a category. We also define the zero morphism (`0`) between two finitely presented modules, the sum (`phi + psi`) of two morphisms and the negation (`- phi`) of a morphism, respectively given by the zero matrix, the sum and the negation of the underlying matrices. As everything is based on matrices, it is straightforward to prove using the divisibility theory of matrices that this is a pre-additive category (i.e. the hom-sets form abelian groups).

However, morphisms are not uniquely represented by an element of `'Mor(M,N)`, but it is possible to test if two morphisms $\varphi\ \psi : M \to N$ are equal by checking if $\varphi - \psi$ is zero modulo the relations of N.

```
Definition eqmor (M N : fpmodule R) (phi psi : 'Mor(M,N)) :=
  pres N %| phi%:m - psi%:m.

(* With notation: *)
(* phi %= psi = eqmor phi psi *)
```

As this is an equivalence relation it would be natural to either use Coq setoid mechanism [4,24] or quotients [6] in order to avoid applying symmetry, transitivity and compatibility with operators (e.g. addition and multiplication) by hand where it would be more natural to use rewriting. We have begun to rewrite the library with quotients as we would get a set of morphisms (instead of a setoid), which is closer to the standard category theoretic notion.

3 Monomorphisms, epimorphisms and operations on morphisms

A monomorphism is a morphism $\varphi : B \to C$ such that whenever there are $\psi_1,\ \psi_2 : A \to B$ with $\psi_1\varphi = \psi_2\varphi$ then $\psi_1 = \psi_2$. When working in pre-additive categories the condition can be simplified to, whenever $\psi\varphi = 0$ then $\psi = 0$.

```
Definition is_mono (M N : fpmodule R) (phi : 'Mor(M,N)) :=
  forall (P : fpmodule R) (psi : 'Mor(P, M)),
    psi ** phi %= 0 -> psi %= 0.

Record monomorphism_of := Monomorphism {
  morphism_of_mono :> 'Mor(M, N);
  _ : kernel morphism_of_mono %= 0
}.

(* With notation: *)
(* 'Mono(M,N) = monomorphism_of M N *)
```

It is convenient to think of monomorphisms $B \to C$ as defining B as a subobject of C, so a monomorphism $\varphi : M \to N$ can be thought as a representation of a submodule M of N. However, submodules are not uniquely represented by monomorphisms even up to equality (`%=`) of morphism. Indeed, multiple monomorphisms with different sources can represent the same submodule. Although "representing the same submodule" is decidable in our theory, introducing the notion of submodules is not mandatory to develop it and we chose not to.

Intuitively monomorphisms correspond to injective morphisms (i.e. with kernel being zero) we would expect that for finitely presented modules this these two notions coincide. The dual notion of monomorphisms are epimorphisms which intuitively correspond to surjective morphisms (i.e. with cokernel being zero). A priori it is not clear that these notions coincide for finitely presented modules. The goal of this section is to clarify this by defining when a finitely presented module is zero, how to compute kernels and cokernels, and the correspondence between injective (resp. surjective) functions and mono- (resp. epi-) morphisms.

3.1 Testing if finitely presented modules are zero

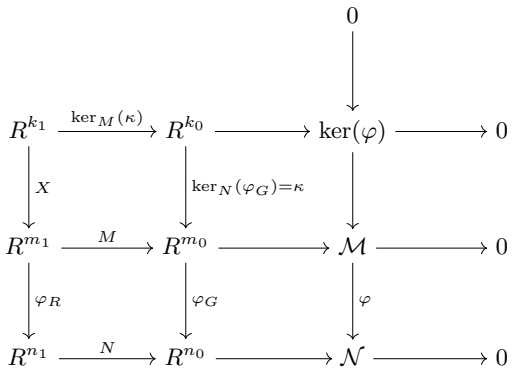
As a finitely presented module is the cokernel of a presentation matrix we have that if the presentation matrix of a module is the identity matrix of dimension $n \times n$ the module is isomorphic to n copies of the zero module. Now consider the following diagram:

$$\begin{array}{ccccccc} R^{m_0} & \xrightarrow{I_{m_0}} & R^{m_0} & \longrightarrow & 0^{m_0} & \longrightarrow & 0 \\ \downarrow{\scriptstyle X} & & \downarrow{\scriptstyle I_{m_0}} & & \downarrow & & \\ R^{m_1} & \xrightarrow{M} & R^{m_0} & \longrightarrow & \mathcal{M} & \longrightarrow & 0 \end{array}$$

which commutes if $\exists X, XM = I_{m_0}$, i.e. when $M \mid I_{m_0}$. Hence this gives a condition that can be tested in order to see if a module is zero or not.

3.2 Computing the kernel of a morphism

In order to compute the kernel of a morphism the key observation is that there is a commutative diagram:



It is easy to see that κ is a monomorphism, which means that the kernel is a submodule of M as expected. In Coq this is easy to define:

```
Definition kernel (M N : fpmodule R) (phi : 'Mor(M,N)) :=
  mor_of_mx ((pres N).-ker phi).
```

Where `mor_of_mx` takes a matrix K with as many columns as M and builds a morphism from $ker_M(K)$ to M. Using this it is possible to test if a morphism is injective:

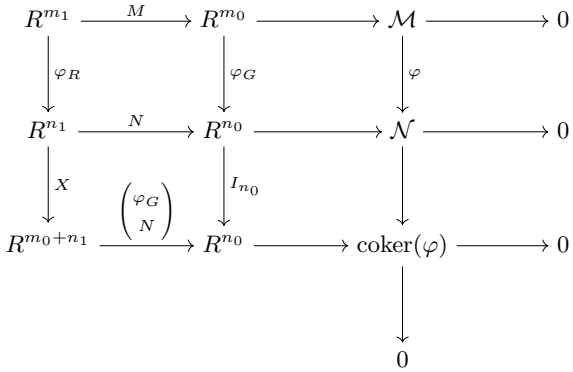
```
Definition injm (M N : fpmodule R) (phi : 'Mor(M,N)) :=
  kernel phi %= 0.
```

We have proved that a morphism is injective if and only if it is a monomorphism:

```
Lemma monoP (M N : fpmodule R) (phi : 'Mor(M,N)) :
  reflect (is_mono phi) (injm phi).
```

3.3 Computing the cokernel of a morphism

The presentation of the cokernel of a morphism can also be found by looking at a commutative diagram:



Note that the canonical surjection onto the cokernel is given by the identity matrix. The fact that this is a morphism is clear as X may be $\begin{pmatrix} 0 & I_{n_1} \end{pmatrix}$. However, before defining this we can define the more general operation of quotienting a module by the image of a morphism by stacking matrices:

```
Definition quot_by (M N : fpmodule R) (phi : 'Mor(M, N)) :=
  FPModule (col_mx (pres N) phi)
```

Now the cokernel is the canonical surjection from `N` onto `quot_by phi`. Since it maps each generator to itself, the underlying matrix is the identity matrix.

```
Definition coker : 'Mor(N, quot_by) :=
  Morphism1 (dvd_quot_mx (dvdmx_refl _)).
```

We can now test if a morphism is surjective by comparing the cokernel of phi with the zero morphism, which coincides with epimorphisms:

```
Definition surjm (M N : fpmodule R) (phi : 'Mor(M,N)) :=
  coker phi %= 0.

Lemma epiP (M N : fpmodule R) (phi : 'Mor(M,N)) :
  reflect (is_epi phi) (surjm phi).
```

Now we have algorithms computing both if a morphism is injective and surjective we can easily test if it is an isomorphism:

```
Definition isom (M N : fpmodule R) (phi : 'Mor(M,N)) :=
  injm phi && surjm phi.
```

A natural question to ask is if we get an inverse from this notion of isomorphism. In order to show this we have introduced the notion of `isomorphisms` that take two morphisms and express that they are mutual inverse of each other, in the sense that given $\varphi : M \to N$ and $\psi : N \to M$ then $\varphi\psi = 1_M$ modulo the relations in M. Using this we have proved:

```
Lemma isoP (M N : fpmodule R) (phi : 'Mor(M,N)) :
  reflect (exists psi, isomorphisms psi) (isom phi).
```

Hence isomorphisms are precisely the morphisms that are both mono and epi. Note that this does not mean that we can decide if two modules are isomorphic, what we can do is test if a given morphism is an isomorphism or not.

3.4 Computing homology

The homology at $\mathcal{N}$ in the sequence:

$$\mathcal{M} \xrightarrow{\varphi} \mathcal{N} \xrightarrow{\psi} \mathcal{K}$$

where $\varphi\psi = 0$ is defined as $\ker(\psi)/\mathrm{im}(\varphi)$. As $\varphi\psi = 0$ we have that $\mathrm{im}(\varphi) \subset \ker(\psi)$, in particular this means that we have an injective map $\iota : \mathrm{im}(\varphi) \to \ker(\psi)$. The homology at $\mathcal{N}$ is the cokernel of this map. In our formalization, submodules are represented by morphisms, so we can write:

```
Hypothesis mul_phi_psi (M N K : fpmodule R) (phi : 'Mor(M,N))
  (psi : 'Mor(N,K)) : phi ** psi %= 0.

Definition homology (M N K : fpmodule R) (phi : 'Mor(M,N))
  (psi : 'Mor(N,K)) := kernel psi %/ phi.
```

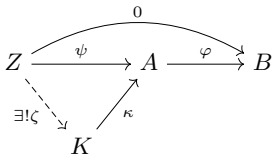
Where `%/` is a notation for taking the quotient of a monomorphism by a morphism with the same target.

In the next section, we show that these operations indeed satisfy the axioms of abelian categories.

4 Abelian categories

As mentioned in the end of Sect. 2 the collection of morphisms between two finitely presented modules forms an abelian group. This means that the category of finitely presented modules and their morphisms is a **pre-additive category**. It is easy to show that the `dsum` construction provides both a product and coproduct. This means that the category is also **additive**.

In order to show that we have a **pre-abelian** category we need to show that morphisms have both a kernel and cokernel in the sense of category theory. A morphism $\varphi : A \to B$ has a kernel $\kappa : K \to A$ if $\kappa\varphi = 0$ and for all $\psi : Z \to A$ with $\psi\varphi = 0$ the following diagram commutes:



This means that any morphism with $\psi\varphi = 0$ factors uniquely through the kernel κ. The dual statement for cokernels state that any morphism ψ with $\varphi\psi = 0$ factors uniquely through the cokernel of φ. The specification of the kernel can be written.

```
Definition is_kernel (M N K : fpmodule R) (phi : 'Mor(M,N))
  (k : 'Mor(K,M)) :=
  (k ** phi %= 0) *
  forall L (psi : 'Mor(L,M)),
    reflect (exists Y, Y ** k %= psi) (psi ** phi %= 0).
```

We have proved that our definition of kernel satisfies this specification:

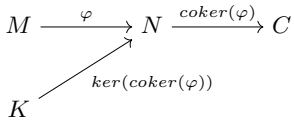
```
Lemma kernelP (M N : fpmodule R) (phi : 'Mor(M,N)) :
  is_kernel phi (kernel phi).
```

We have also proved the dual statement for cokernels. The only properties left in order to have an **abelian category** is that every monomorphism (resp. epimorphism) is normal which means that it is the kernel (resp. cokernel) of some morphism. We have shown that if φ is a monomorphism then its cokernel satisfies the specification of kernels:

```
Lemma mono_ker (M N : fpmodule R) (phi : 'Mono(M,N)) :
  is_kernel (coker phi) phi.
```

This means that φ is a kernel of $coker(\varphi)$ if φ is a monomorphism, hence are all monomorphisms normal. We have also proved the dual statement for epimorphisms which means that we indeed have an abelian category.

It is interesting to note that many presentations of abelian categories are sloppy and say that `phi` is `kernel(coker phi)`, but this is not even well-typed as:



One cannot just subtract φ and ker(coker(φ)) as they have different sources. In order to express this formally we need to exhibit an isomorphism from M and K and insert it in the equation.

However, if we introduced a notion of submodule of M as a quotient of injective morphism from any module N into M by the equivalence relation that identifies $\varphi : N \to M$ and $\varphi' : N' \to M$ if there exists an isomorphism $\psi : N \to N'$ such that $\varphi = \alpha\varphi'$.

5 Smith normal form

As mentioned before, it is in general not possible to decide if two presentations represent isomorphic modules, even when working over coherent strongly discrete rings. When the underlying ring is a field it is possible to represent a finite dimensional vector space in a canonical way as they are determined up to isomorphism by their dimension (i.e. the rank of the underlying matrix) which can be computed by Gaussian elimination [12]. A generalization of this is the class of rings, called *elementary divisor rings* by Kaplansky [19], where any matrix is equivalent[4] to a matrix in Smith normal form:

[4] A matrix M is *equivalent* to a matrix D if there exist invertible matrices P and Q such that $PMQ = D$.

Definition 4. *A matrix is in* Smith normal form *if it is a diagonal matrix of the form:*

$$\begin{pmatrix} d_1 & & 0 & \cdots & \cdots & 0 \\ & \ddots & & & & \vdots \\ 0 & & d_k & 0 & \cdots & 0 \\ \vdots & & 0 & 0 & & \vdots \\ \vdots & & \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & \cdots & 0 \end{pmatrix}$$

where $d_1 \mid d_2 \mid \ldots \mid d_k$.

The connection between elementary divisor rings and finitely presented modules is that the existence of a Smith normal form for the presentation matrix gives us:

$$\begin{array}{ccccccc} R^{m_1} & \xrightarrow{M} & R^{m_0} & \longrightarrow & \mathcal{M} & \longrightarrow & 0 \\ \downarrow{\scriptstyle P^{-1}} & & \downarrow{\scriptstyle Q} & & \downarrow{\scriptstyle \varphi} & & \\ R^{m_1} & \xrightarrow{D} & R^{m_0} & \longrightarrow & \mathcal{D} & \longrightarrow & 0 \end{array}$$

Now φ is an isomorphism as P and Q are invertible. In order to represent this in Coq we need to represent diagonal matrices, we use the function `diag_mx_seq`. It is a function that takes two numbers m and n and a list and returns a matrix of type `'M[R]_(m,n)` where the elements of the diagonal are the elements of the list. It is defined as follows:

```
Definition diag_mx_seq m n (s : seq R) :=
  \matrix_(i < m, j < n) (s`_i *+ (i == j :> nat)).
```

This means that the i^{th} diagonal element of the matrix is the i^{th} element of the list and the rest are zero. Now if M is a matrix, our algorithm for computing the Smith normal form should return a list `s` and two matrices P and Q such that:

1. `s` is sorted by division and its length is less than m and n,
2. `P *m M *m Q = diag_mx_seq m n s` and
3. `P` and `Q` are invertible.

If we assume that R is coherent and strongly discrete we can consider finitely presented modules over R. As `P` is invertible it is obvious that `Q` defines a morphism from `M` to `diag_mx_seq m n s`. Also `P^-1` defines a morphism in the other direction that is inverse to `P` which means that `M` and `diag_mx_seq m n s` are isomorphic.

Bézout domains

We now assume that all rings have explicit divisibility, that is, we can decide is $a \mid b$ and if this is the case produce x such that $b = xa$.

Definition 5. *An integral domain R is a* ***Bézout domain*** *if every finitely generated ideal is principal (generated by a single element).*

This is equivalent to requiring that R has a GCD operation and a function computing the elements of the Bézout identity; this means that given a and b one can compute x and y such that $xa + by$ is associate[5] to $gcd(a, b)$.

We have formalized a proof that Bézout domains of Krull dimension less than or equal to 1 (in particular principal ideal domains like $\mathbb{Z}$ and $k[x]$ with k a field) are elementary divisor rings, however as this paper is concerned with finitely presented modules we do not go into the details of this proof here. The reason we restrict our attention to rings of Krull dimension less than or equal to 1 is that it is still an open problem whether all Bézout domains are elementary divisor domains or not [21].

Combining this with finitely presented modules we get a constructive generalization of the classification theorem of finitely generated modules over principal ideal domains which states that any finitely presented R-module $\mathcal{M}$ over a principal ideal domain R can be decomposed into a direct sum of a free module and cyclic modules, that is, there exists $n \in \mathbb{N}$ and elements $d_1, \ldots, d_k \in R$ such that:

$$\mathcal{M} \simeq R^n \oplus R/(d_1) \oplus \cdots \oplus R/(d_k)$$

with the additional property that $d_1 \mid d_2 \mid \cdots \mid d_k$.

In [5], it is proved that the Smith normal form is unique up to multiplication of units. This means that for any matrix M equivalent to a diagonal matrix D in Smith normal form, each of the diagonal elements of the Smith normal form of M will be associate to the corresponding diagonal element in D. This implies that the decomposition of finitely presented modules over elementary divisor rings is unique up to multiplication by units.

6 Conclusions and future work

In this paper we have presented a formalization of the category of finitely presented modules over coherent and strongly discrete rings and shown that it is an abelian category. The fact that we can represent everything with matrices makes is possible for us to reuse basic results on these when building the abstraction layer of modules on top. The division theory of matrices makes it straightforward for us to do reasoning modulo a set of relations.

It is not only interesting that we have an abelian category because it provides us with a setting to do homological algebra, but also because it is proved in [9] that in order to show that abelian groups (and hence the category of R-modules) form an abelian category in Coq one needs the principle of unique choice. So what we have established here is that

[5] a and b are *associates* if $a \mid b$ and $b \mid a$ or equivalently that there exists a unit $u \in R$ such that $a = bu$.

this is not necessary for the category of finitely presented modules over coherent strongly discrete rings.

In Homotopy Type Theory [25] there is a distinction between pre-categories and univalent categories (just called categories in [1]). A *pre-category* is a category where the collection of morphisms forms a set in the sense of homotopy type theory, that is, they satisfy the uniqueness of identity proofs principle. Our category of presented modules satisfy the uniqueness of morphism equivalence (`phi %= psi`) proofs (by Hedberg's theorem [16]), but morphisms form a setoid instead of a set. If we quotiented morphisms by the equivalence relation on morphisms we would get a set, and thus our category of finitely presented modules would become a pre-category.

A *univalent category* on the other hand is a pre-category where the equality of objects coincides with isomorphisms. As we have shown that for elementary divisor rings there is a way to decide isomorphism, we could also get a univalent category by quotienting modules by isomorphisms. It would be interesting to develop these ideas further and define the notion of univalent abelian category and study its properties. Note that in Homotopy Type Theory, it is no longer necessary to have the decidability of the equivalence relation to form the quotient, so we would not need to be in an elementary divisor ring to get a univalent category.

Since we have shown that we have an abelian category it would now be very interesting to formally study more complex constructions from homological algebra. It should for instance be straightforward to compute resolutions of modules. We can then define define Hom and tensor functors in order to get derived functors like Tor and Ext. It would also be interesting to define graded objects like chain complexes and graded finitely presented modules, and prove that they also are abelian categories.

Acknowledgments: The authors are grateful to Bassel Mannaa for his comments on early versions of the paper.

References

1. B. Ahrens, C. Kapulkin, and M. Shulman. Univalent categories and the Rezk completion, 2013. Preprint. `http://arxiv.org/abs/1303.0584`.
2. M. Barakat and M. Lange-Hegermann. An axiomatic setup for algorithmic homological algebra and an alternative approach to localization. *J. Algebra Appl.*, 10(2):269–293, 2011.
3. M. Barakat and D. Robertz. HOMALG – A Meta-Package for Homological Algebra. *J. Algebra Appl.*, 7(3):299–317, 2008.
4. G. Barthe, V. Capretta, and O. Pons. Setoids in type theory. *Journal of Functional Programming*, 13(2):261–293, 2003.
5. G. Cano and M. Dénès. Matrices à blocs et en forme canonique. In *JFLA - Journées francophones des langages applicatifs*, 2013.
6. C. Cohen. Pragmatic Quotient Types in Coq. In *Interactive Theorem Proving*, volume 7998 of *LNCS*, pages 213–228. 2013.

7. Coq development team. The Coq Proof Assistant Reference Manual, version 8.4. Technical report, Inria, 2012.
8. T. Coquand, A. Mörtberg, and V. Siles. Coherent and strongly discrete rings in type theory. CPP'12, pages 273–288, 2012.
9. T. Coquand and A. Spiwack. Towards constructive homological algebra in type theory. Calculemus '07 / MKM '07, pages 40–54, 2007.
10. W. Decker and C. Lossen. *Computing in Algebraic Geometry: A Quick Start using SINGULAR*. Springer, 2006.
11. F. Garillot, G. Gonthier, A. Mahboubi, and L. Rideau. Packaging mathematical structures. In *TPHOLs'09*, volume 5674 of *LNCS*, pages 327–342, 2009.
12. G. Gonthier. Point-Free, Set-Free concrete linear algebra. In *ITP'11*, volume 6898 of *LNCS*, pages 103–118, 2011.
13. G. Gonthier and A. Mahboubi. A Small Scale Reflection Extension for the Coq system. Technical report, Microsoft Research INRIA, 2009.
14. G.-M. Greuel and G. Pfister. *A Singular Introduction to Commutative Algebra*. 2nd edition, 2007.
15. A. Hatcher. *Algebraic Topology*. Cambridge University Press, 1st edition, 2001.
16. M. Hedberg. A Coherence Theorem for Martin-Löf's Type Theory. *Journal of Functional Programming*, 8(4):413–436, 1998.
17. J. Heras, T. Coquand, A. Mörtberg, and V. Siles. Computing persistent homology within Coq/SSReflect. *ACM Transactions on Computational Logic*, 14(4):26, 2013.
18. J. Heras, M. Dénès, G. Mata, A. Mörtberg, M. Poza, and V. Siles. Towards a certified computation of homology groups for digital images. In *CTIC'12*, volume 7309 of *LNCS*, pages 49–57, 2012.
19. I. Kaplansky. Elementary divisors and modules. *Transactions of the American Mathematical Society*, 66:464–491, 1949.
20. H. Lombardi and C. Quitté. *Algèbre commutative, Méthodes constructives: Modules projectifs de type fini*. Calvage et Mounet, 2011.
21. D. Lorenzini. On Bézout Domains. `http://www.math.uga.edu/~lorenz/Bezout.pdf`.
22. R. Mines, F. Richman, and W. Ruitenburg. *A Course in Constructive Algebra*. Springer-Verlag, 1988.
23. H. Poincaré. Analysis situs. *Journal de l'École Polytechnique*, 1:1–123, 1895.
24. M. Sozeau. A new look at generalized rewriting in type theory. *Journal of Formalized Reasoning*, 2(1):41–62, 2009.
25. The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study, 2013. `http://homotopytypetheory.org/book/`.