Computations of the structure of module categories using

FD Applet

https://fd-applet.dt.r.appspot.com

Haruhisa Enomoto (Osaka Metropolitan Univ.)

Contents

1. Demo.

2. Basics on String Algebras

3. Algorithms for computable alg

4. Lean Theorem Prover

2. Basics on String Algebras

(**Rem**
Works for
special biserial alg)

Def A string algebra is a fin. dim quiver alg kQ/I s.t.

(1) $\forall i \in Q$, $\#\{i \to\} \leq 2$,
$\#\{\to i\} \leq 2$

(2) $\forall$
$$\begin{tikzcd} & & {} \\ {} \arrow[r, "a"] & \bullet \arrow[ur, "x"] \arrow[dr, "y"'] & \\ & & {} \end{tikzcd}$$
, $ax = 0$ or $ay = 0$
$(\in I)$ $(\in I)$

$(2)^{op}$ $\forall$
$$\begin{tikzcd} {} \arrow[dr, "x"] & & \\ & \bullet \arrow[r, "b"] & {} \\ {} \arrow[ur, "y"'] & & \end{tikzcd}$$
, $xb = 0$ or $yb = 0$

(3) I is generated by paths (= monomials)

Ex

- Nakayama alg. $\cdot \to \cdot \to \cdot \to \cdot \to \cdot$
$$\begin{tikzcd} \cdot \arrow[r] & \cdot \arrow[d] \\ \cdot \arrow[u] & \cdot \arrow[l] \end{tikzcd}$$
- type A path alg. $\cdot \to \cdot \leftarrow \cdot \leftarrow \cdot \to \cdot \to \cdot \leftarrow \cdot$
-
$$\begin{tikzcd} 1 \arrow[dr] & & 4 \\ & 3 \arrow[ur] \arrow[dr] & \\ 2 \arrow[ur] & & 5 \end{tikzcd}$$
-
$$\begin{tikzcd} 1 \arrow[r, "a"] & 2 \arrow[loop above, "b"] \end{tikzcd} \Big/ \langle ab, b^5 \rangle$$
$$\begin{tikzcd} \arrow[dr, "a"] & & \\ & 2 \arrow[r, "b"] & {} \\ \arrow[ur, "b"'] & & \end{tikzcd}$$

etc.

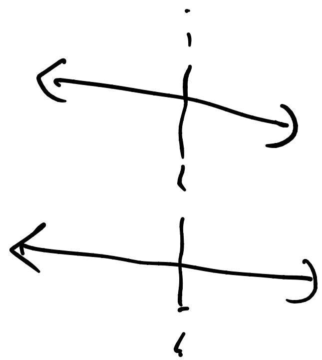
$$\begin{tikzcd} & & 3 \\ 1 \arrow[r] & 2 \arrow[ur] \arrow[dr] & \\ & & 4 \end{tikzcd}$$
: not string alg

Classification of indecs over string alg

Thm Every indec module over a string alg is either of the following:

(1) String module

(2) Band module



Combinatorics

string

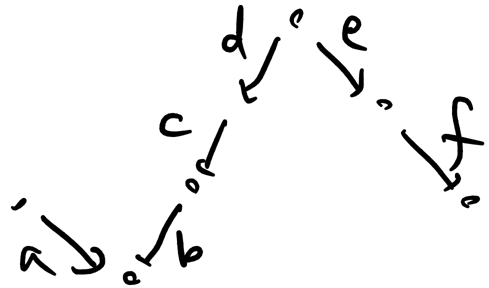
band

Moreover, representation-finite

$\Longleftrightarrow$ No band modules

String module

is a module looks like:



FD Applet

a * !b * !c * !d * e * f

$a\, b^{-1} c^{-1} d^{-1} e\, f$: string

Def String of kQ/I is

a word $l_1 \cdots l_n$ s.t.

- $l_i = a_i$: arrow or a_i^{-1} : inverse of arrow
- $t(l_i) = s(l_{i+1})$ $\forall i$
- does NOT contain "aa^{-1}", "$a^{-1}a$", and any relations.

String module

$$Q: \; 1 \xrightarrow{a} 2 \xrightarrow{b} 3 \xleftarrow{c} 4$$

$a\, b\, c^{-1}$

$\rightsquigarrow$ draw

$$\begin{tikzcd}[column sep=small, row sep=small]
1 \arrow[dr, "a"'] & & & \\
& 2 \arrow[dr, "b"'] & & 4 \arrow[dl, "c"] \\
& & 3 &
\end{tikzcd}$$

a $\qquad\qquad$ a^{-1}

$\rightsquigarrow$ View this as rep of Q!

(Put "k" on each vtx of diagram,
regard it as basis at the label,
and define actions of arrows by the diagram.)

Ex $\quad Q: \quad 1 \underset{b}{\overset{a}{\rightrightarrows}} 2$

$$ab^{-1} \rightsquigarrow \begin{tikzcd} 1 \arrow[dr, "a"'] & & 1 \arrow[dl, "b"] \\ & 2 & \end{tikzcd}$$

$$\rightsquigarrow \quad \text{vtx } 1 \;\; k \oplus k \underset{b}{\overset{a}{\rightrightarrows}} k \;\; \text{vtx } 2.$$

- No linear alg needed!
- Exercise in computer programming!

Obs For a given kQ/I, it's computable to list all string.

Rep thy $\rightsquigarrow$	Combinatorics
String module $\longleftrightarrow$	string.

Band   left-right

A band is an infinite periodic string.

Ex $1 \underset{b}{\overset{a}{\rightrightarrows}} 2 \quad \rightsquigarrow \quad \cdots ab^{-1}ab^{-1}ab^{-1}\cdots$: band

Band + indec $k[x,x^{-1}]$-module (Jordan if $k=\bar{k}$)

$\rightsquigarrow$ "Band module" (details omitted).

Obs For a given kQ/I,

it's computable to check

whether there are NO bands

($\Leftrightarrow$ rep-fin)

In the rest,

$A = kQ/I$: f.d. string algebra

w : string $\rightsquigarrow$ $M(w)$: string module

Assume No Bands i.e. A : rep-fin.

<u>Thm</u> If A has no bands,

$$\{\text{indecs in } \mathrm{mod}A\} \xleftrightarrow[\text{1-1}]{} \{\text{strings}\} \big/ w \sim w^{-1}$$

Rep theory - - - - - Combinatorics

Want to describe categorical str of $\mathrm{mod}A$ using only combinatorics of strings.

Known results

1. AR quiver is computable. [Butler-Ringel]

2. $\dim_k \mathrm{Hom}_A(X, Y)$ is computable

$\mathrm{Hom}_A(M(w_1), M(w_2))$ has basis [Crawley-Boevey]

$\{ w_1 \twoheadrightarrow w \hookrightarrow w_2 \}$

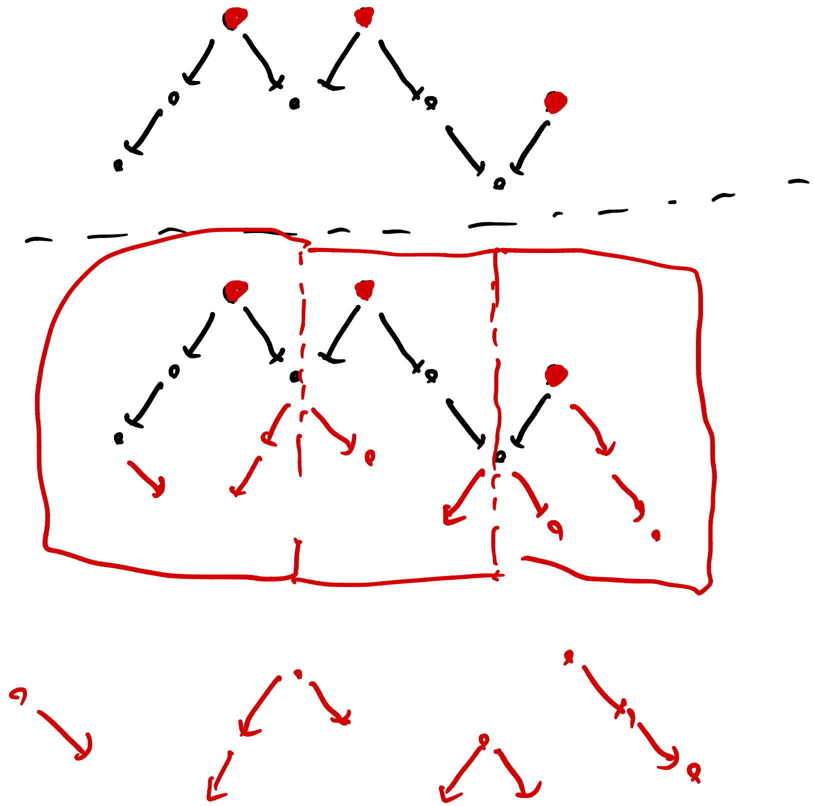
3. Proj cover (inj hull) is computable

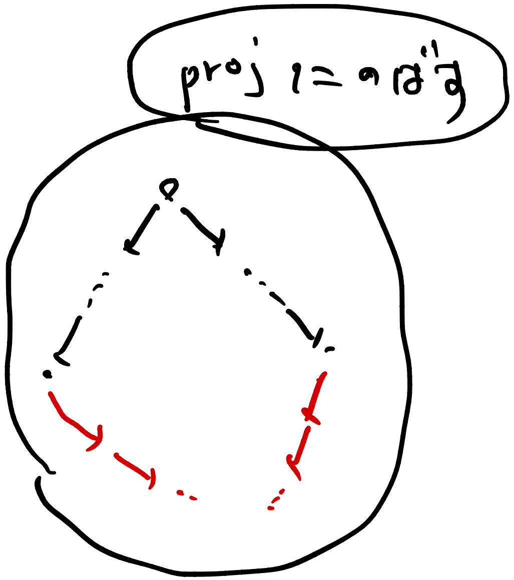
4. Syzygy (cosyzygy) is computable

[Allen, Syzygies of string modules for special biserial algebras]

Sketch

w :





proj cover

proj

ker

: syzygy

3. Algorithm for

Computable alg

"**Def**" A f.d. alg A is computable

if A is rep-fin and satisfies

(C1) ind (mod A) is computable

i.e., ind (mod A) $\underset{1-1}{\longleftrightarrow}$ { combinatorial objs }

(C2) $\forall X, Y \in$ ind (mod A),

$\dim_K \mathrm{Hom}_A(X, Y)$ is computable.

(C3) $\forall X \in$ ind (mod A),

Both $\begin{pmatrix} \text{proj cover} \\ \text{syzygy} \end{pmatrix}$ are computable

(C4) All AR seq are computable

Prop If A is computable, then so are the following for $\forall\, X, Y \in \mathrm{mod}\, A$

(1) $\dim_k \mathrm{Ext}_A^i(X, Y)$ for every $i \geq 0$

(2) Whether " $\mathrm{Ext}_A^i(X, Y) = 0 \;\; \forall i > 0$ " or not.

computable

(1) $\mathrm{Ext}_A^i(X, Y) = \mathrm{Ext}_A^1(\underline{\Omega^{i-1} X}, Y)$, so let $i = 1$.

$$0 \to \Omega X \to P \to X \to 0$$

$$\rightsquigarrow \quad 0 \to (X, Y) \to (P, Y) \to (\Omega X, Y) \to \mathrm{Ext}^1(X, Y) \to 0$$

$\rightsquigarrow$ Count dimension!

(2) $\Omega^* X := \{ \Omega^i X \mid i \geq 0 \}$ is computable

$\rightsquigarrow$ Whether " $\mathrm{Ext}_A^1(M, Y) = 0 \;\; \forall M \in \Omega^* X$ " is computable

$\square$

Cor If A is computable, then so are the following.

(1) $\forall X$, $\mathrm{pd}_A X$, $\mathrm{id}_A X$,

(2) $\mathrm{gl.dim}\, A$

(3) The set of all

(i) (partial) classical tilting modules

(ii) Miyashita tilting modules

(iii) Wakamatsu tilting = semi-dualizing modules $\lrcorner$

∵ (3) They are characterized by

s.t. (i) $\mathrm{pd}\, T \leq 1$, $\mathrm{Ext}^1_A(T,T) = 0$, $|T| = |A|$

My Result (ii) $\mathrm{pd}\, T < \infty$, $\mathrm{Ext}^{>0}_A(T,T) = 0$, $|T| = |A|$

(iii) $\mathrm{Ext}^{>0}_A(T,T) = 0$, $|T| = |A|$ $\square$

Modules and subcats in τ-tilting theory

Semibrick

Assume A : computable / alg. cl. field

- $B \in \operatorname{ind}(\operatorname{mod} A)$: brick
  $\iff \dim_K \operatorname{Hom}_A(B, B) = 1$

  $\therefore$ $\operatorname{brick} A := \{\text{bricks in } \operatorname{mod} A\}$: computable

- Semibrick : set of pair-wise Hom-ortho bricks

  $\therefore$ $\operatorname{sbrick} A := \{\text{semibricks} \text{ ——— }\}$: computable

Torsion pairs

__Fact__ For $M \in \operatorname{mod} A$,

$({}^{\perp}(M^{\perp}), M^{\perp})$: torsion pair

$\;\;\,\parallel$

$T(M)$: the smallest tors containing M.

$T(M)$: computable

Fact 2. (If A: τ-tilt. fin.)

Every torsion pair is of the form
$(\mathsf{T}(S), S^{\perp})$ for a semibrick S.

$\rightsquigarrow$ {tors pairs} : computable

Below are computable bijections in τ-tilt. theory

$$
\begin{tikzcd}
\operatorname{tors} A \arrow[d, "\text{Ext-proj}"'] & \operatorname{sbrick} A \arrow[l, "\mathsf{T}(-)"'] \arrow[r, "(-)^{\perp}"] \arrow[d, "\psi"'] \arrow[dr, "\varphi"'] & \operatorname{torf} A \\
s\tau\text{-tilt}\, A & \operatorname{wide} A & \operatorname{sbrick} A \arrow[u, "\mathsf{F}(-)"']
\end{tikzcd}
$$

φ : s.t. $S \oplus \varphi(S)[1]$: 2-simple minded collection.

$\psi(S) := \mathsf{T}(S) \cap \mathsf{F}(S)$: wide for S : sbrick

Other subcats

(1) ICE-closed subcats (K) $(:\Longleftrightarrow$ closed under Image-Coker-Ext$)$

$\|$

tors in some wide. (torf)

(2) IE-closed subcats $= \mathcal{T} \cap \mathcal{F}$ $\exists$ $\mathcal{T}$: tors, $\mathcal{F}$: torf

(3) Resolving subcats $\mathcal{X}$ $\longleftrightarrow$ hereditary cotorsion pair

$\Longleftrightarrow$ $\mathcal{X} = {}^{\perp_{>0}}(\mathcal{E})$ $\exists$ $\mathcal{E}$: subcat

(4) Subcats $\mathcal{X}$ closed under ext, summands AND contains A

$\longleftrightarrow$ cotorsion pair.

$\Longleftrightarrow$ $\mathcal{X} = {}^{\perp_1}(\mathcal{E})$ $\exists$ $\mathcal{E}$: subcat.

Questions

(1) Assume A is computable.
Are the following subcategories computable?

(i) closed under extensions & summands
(ii) extensions & kernels
(iii) kernels & cokernels
(iv) images
(v) submodules

⋱ etc

(2) Is Dynkin path alg computable?

(3) Can we make "computable" more rigorous?

(Maybe computation theory needed?

(Turing Machine etc))

(4) How about "complexity" of actual computation?

(Many problems reduce to graph clique problem.)

(Obtaining resolving subset, tors → sbrick, ...

is slow, ..., efficient or optimal algorithm?

4. Lean Theorem Prover

- コンピュータのプログラムとして数学の証明をかける（エラー無し ⇒ 証明は正しい！）
- 学部数学程度はすでに自由に使える（数学者が趣味で大勢協力している）
- 「仮定を一般化して証明なりたつか？」等がすぐ分かる
- 「AI様」に数学を教えることができる！
- 査読プロセスが簡単に？

伝えたいこと

- 現在 Lean は純粋数学者で少しずつ認識されている（Scholze 氏…）
- しかし「環」は多くが可換を仮定されている ⇝ 我々の力が必要!!

（局所環・中山，… は可換にしかない
アーベル圏・三角圏あるが完全圏・ET圏ない，etc）

- 9/3 の「数学系のためのLean勉強会」
https://haruhisa-enomoto.github.io/lean-math-workshop/
の教材で入門できるので，Help Us!!
（群の def から準同型定理までの教材をがんばって作ったので遊んでください．）
質問は何でもメールしてください