Concurrent Sylvester Solver

Joshua Maglione

Universität Bielefeld
`jmaglione@math.uni-bielefeld.de`

James B. Wilson

Colorado State University
`james.wilson@colostate.edu`

Version 2.2
January 24, 2019

Contents

CHAPTER 1

Introduction

CHAPTER 2

Invariants for bilinear tensors

The following intrinsics are specialized for tensors of valence 3, but equivalent intrinsics for general tensors are presented in the proceeding subsection.

`AdjointAlgebra(t) : TenSpcElt -> AlgMat`

Returns the adjoint $*$-algebra of the given Hermitian bilinear map t, represented on $\mathrm{End}(U_2)$. This is using algorithms from StarAlge, see the `AdjointAlgebra` intrinsic in [?BW:StarAlge]. If the current version of StarAlge is not attached, the default Magma version will be used instead.

Example 2.1. `AdjointAlge`

Given the context of [?BW:isometry], we construct a tensor from a p-group and compute its adjoint algebra.

```
> G := SmallGroup(3^7, 7000);
> t := pCentralTensor(G);
> t;
Tensor of valence 3, U2 x U1 >-> U0
U2 : Full Vector space of degree 4 over GF(3)
U1 : Full Vector space of degree 4 over GF(3)
U0 : Full Vector space of degree 3 over GF(3)
```

Unlike other intrinsics that compute invariants of tensors, `AdjointAlgebra` exploits the fact that t is Hermitian so that the adjoint algebra is faithfully represented on $\mathrm{End}(U_2) = \mathrm{End}(U_1)$.

```
> A := AdjointAlgebra(t);
> A;
Matrix Algebra of degree 4 and dimension 4 with 4 generators over GF(3)
> A.1;
[1 0 0 0]
[0 0 0 0]
[0 0 0 0]
[0 0 0 1]
```

Because A is constructed from algorithms in StarAlge, we can apply other algorithms from that package specifically dealing with the involution on A. See StarAlge [?BW:StarAlge] for descriptions of the intrinsics.

```
> RecognizeStarAlgebra(A);
true
> SimpleParameters(A);
[ <"symplectic", 2, 3> ]
> Star(A);
Mapping from: AlgMat: A to AlgMat: A given by a rule [no inverse]
```

`LeftNucleus(t) : TenSpcElt -> AlgMat`
    `op : BoolElt : false`

Returns the left nucleus of the bilinear map t as a subalgebra of $\mathrm{End}(U_2)\times\mathrm{End}(U_0)$. In previous versions of TensorSpace (and eMAGma), the left nucleus was returned as a subalgebra of $\mathrm{End}(U_2)^\circ \times \mathrm{End}(U_0)^\circ$. To enable this, set the optional arugment `op` to `true`.

`MidNucleus(t) : TenSpcElt -> AlgMat`

Returns the mid nucleus of the bilinear map t as a subalgebra of $\mathrm{End}(U_2) \times \mathrm{End}(U_1)^\circ$.

`RightNucleus(t) : TenSpcElt -> AlgMat`

Returns the right nucleus of the bilinear map t as a subalgebra of $\mathrm{End}(U_1) \times \mathrm{End}(U_0)$.

Example 2.2. `GoingNuclear`

We will verify a theorem from [?FMW:densors] and [?Wilson:LMR]: all the nuclei of a tensor embed into the derivation algebra. We construct the tensor given by $(3 \times 4 \times 5)$-matrix multiplcation.

```
> K := Rationals();
> A := KMatrixSpace(K, 3, 4);
> B := KMatrixSpace(K, 4, 5);
> C := KMatrixSpace(K, 3, 5);
> F := func< x | x[1]*x[2] >;
> t := Tensor([A, B, C], F);
> t;
Tensor of valence 3, U2 x U1 >-> U0
U2 : Full Vector space of degree 12 over Rational Field
U1 : Full Vector space of degree 20 over Rational Field
U0 : Full Vector space of degree 15 over Rational Field
```

Because matrix multiplication is associative, the left, middle, and right nuclei contain $\mathbb{M}_3(\mathbb{Q})$, $\mathbb{M}_4(\mathbb{Q})$, and $\mathbb{M}_5(\mathbb{Q})$ respectively. In fact, the following computation shows that this is equality.

```
> L := LeftNucleus(t : op := true);
> M := MidNucleus(t);
> R := RightNucleus(t);
> Dimension(L), Dimension(M), Dimension(R);
9 16 25
> D := DerivationAlgebra(t);
> Dimension(D);
49
```

Now we will embed these nuclei into the derivation algebra of t.

```
> Omega := KMatrixSpace(K, 47, 47);
> Z1 := ZeroMatrix(K, 20, 20);
> L_L2, L2 := Induce(L, 2);
> L_L0, L0 := Induce(L, 0);
> embedL := map< L -> Omega | x :->
>     DiagonalJoin(<Transpose(x @ L_L2), Z1, Transpose(x @ L_L0)>) >;
>
> Z0 := ZeroMatrix(K, 15, 15);
> M_M2, M2 := Induce(M, 2);
> M_M1, M1 := Induce(M, 1);
> embedM := map< M -> Omega | x :->
>     DiagonalJoin(<x @ M_M2, -Transpose(x @ M_M1), Z0>) >;
>
> Z2 := ZeroMatrix(K, 12, 12);
> R_R1, R1 := Induce(R, 1);
> R_R0, R0 := Induce(R, 0);
> embedR := map< R -> Omega | x :->
>     DiagonalJoin(<Z2, x @ R_R1, x @ R_R0>) >;
>
> Random(Basis(L)) @ embedL in D;
true
> Random(Basis(M)) @ embedM in D;
true
> Random(Basis(R)) @ embedR in D;
```

```
true
```

CHAPTER 3

Invariants of general multilinear maps

The following functions can be used for general tensors.

```
Centroid(t) : TenSpcElt -> AlgMat
Centroid(t, A) : TenSpcElt, {RngIntElt} -> AlgMat
```

Returns the A-centroid of the tensor as a subalgebra of $\prod_{a\in A} \mathrm{End}(U_a)$, where $A \subseteq [\![\mathfrak{v}]\!]$. If no A is given, it is assumed that $A = [\![\mathfrak{v}]\!]$. If t is contained in a category where coordinates a and b (also contained in A) are fused together, then the corresponding operators on those coordinates will be equal.

Example 3.1. Centroid

The centroid C of a tensor t is the largest ring for which t is C-linear, see [?FMW:densors, Theorem D]. To demonstrate this, we will construct the tensor given by multiplication of the splitting field of $f(x) = x^4 - x^2 - 2$ over $\mathbb{Q}$. However, this field won't explicitly be given with the tensor data.

```
> A := MatrixAlgebra(Rationals(), 4);
> R<x> := PolynomialRing(Rationals());
> F := sub< A | A!1, CompanionMatrix(x^4-x^2-2) >;
> F;
Matrix Algebra of degree 4 with 2 generators over Rational Field
> t := Tensor(F);
> t;
Tensor of valence 3, U2 x U1 >-> U0
U2 : Full Vector space of degree 4 over Rational Field
U1 : Full Vector space of degree 4 over Rational Field
U0 : Full Vector space of degree 4 over Rational Field
```

The centroid is the field $\mathbb{Q}(\sqrt{2}, i)$.

```
> C := Centroid(t);
> C;
Matrix Algebra of degree 12 with 4 generators over Rational Field
> sub< C | C.1 > eq C;
true
> forall{ c : c in Generators(C) | IsInvertible(c) };
true
> IsCommutative(C);
true
> MinimalPolynomial(C.1);
x^4 + 2*x^2 - 8
> Factorization(MinimalPolynomial(C.1));
[
    <x^2 - 2, 1>,
    <x^2 + 4, 1>
]
```

```
DerivationAlgebra(t) : TenSpcElt -> AlgMatLie
DerivationAlgebra(t, A) : TenSpcElt, {RngIntElt} -> AlgMatLie
```

Returns the A-derivation Lie algebra of the tensor as a Lie subalgebra of $\prod_{a\in A}\mathrm{End}(U_a)$, where $A \subseteq [\![\mathfrak{v}]\!]$. If no A is given, it is assumed that $A = [\![\mathfrak{v}]\!]$. If t is contained in a category where coordinates a and b (also contained in A) are fused together, then the corresponding operators on those coordinates will be equal.

```
Nucleus(t, a, b) : TenSpcElt, RngIntElt, RngIntElt -> AlgMat
Nucleus(t, A) : TenSpcElt, SetEnum -> AlgMat
```

Returns the A-nucleus, for $A = \{a, b\}$ $(a \neq b)$, of the tensor as a subalgebra of $\mathrm{End}(U_i) \times \mathrm{End}(U_j)$, where $i = \max(a, b)$ and $j = \min(a, b)$. If $j > 0$, then replace $\mathrm{End}(U_j)$ with $\mathrm{End}(U_j)^{\circ}$. If t is contained in a category where coordinates a and b are fused together, then the corresponding operators on those coordinates will not be forced to be equal.

Example 3.2. `RestrictDerivation`

In a previous example, we embeded the nuclei of a tensor into the derivation algebra. For a tensor $t : U_{\mathfrak{v}} \times \cdots \times U_1 \rightarrowtail U_0$, the derivation algebra is represented in $\Omega = \prod_{a\in[\![\mathfrak{v}]\!]}\mathfrak{gl}(U_a)$. We will restrict the derivation algebra to $\prod_{c\notin\{a,b\}}\mathfrak{gl}(U_c)$ for distinct $a, b \in [\![\mathfrak{v}]\!]$. From [?FMW:densors, Lemma 4.11], the kernel of this restriction is equal to $\mathrm{Nuc}_{\{a,b\}}(t)^{-}$. We will just verify that the dimensions match.

We will construct a tensor given by matrix multiplcation:

$$\mathbb{M}_{3\times 4}(\mathbb{F}_2) \times \mathbb{M}_{4\times 2}(\mathbb{F}_2) \times \mathbb{M}_{2\times 2}(\mathbb{F}_2) \rightarrowtail \mathbb{M}_{3\times 2}(\mathbb{F}_2).$$

```
> A := KMatrixSpace(GF(2), 3, 4);
> B := KMatrixSpace(GF(2), 4, 2);
> C := KMatrixSpace(GF(2), 2, 2);
> D := KMatrixSpace(GF(2), 3, 2);
> trip := func< x | x[1]*x[2]*x[3] >;
> t := Tensor([A, B, C, D], trip);
> t;
Tensor of valence 4, U3 x U2 x U1 >-> U0
U3 : Full Vector space of degree 12 over GF(2)
U2 : Full Vector space of degree 8 over GF(2)
U1 : Full Vector space of degree 4 over GF(2)
U0 : Full Vector space of degree 6 over GF(2)
```

Now we will compute the derivation algebra of `t`. We choose $a = 3$ and $b = 2$, so the $\{3, 2\}$-nucleus is $\mathbb{M}_{4\times 4}(\mathbb{F}_2)$.

```
> D := DerivationAlgebra(t);
> Dimension(D);
32
> N32 := Nucleus(t, 3, 2);
> N32;
Matrix Algebra of degree 20 with 16 generators over GF(2)
```

To construct the restriction of $\mathrm{Der}(t)$ into $\mathfrak{gl}(U_1) \times \mathfrak{gl}(U_0)$ we will use the `Induce` function.

```
> Omega_10 := KMatrixSpace(GF(2), 10, 10);
> D_vs := sub< KMatrixSpace(GF(2), 30, 30) | Basis(D) >;
> pi1, D1 := Induce(D, 1);
> pi0, D0 := Induce(D, 0);
> res := hom< D_vs -> Omega_10 |
>     [<x, DiagonalJoin(x @ pi1, x @ pi0)> : x in Basis(D)] >;
> res;
Mapping from: ModMatFld: D_vs to ModMatFld: Omega_10
> Kernel(res);
KMatrixSpace of 30 by 30 matrices and dimension 16 over GF(2)
```

`SelfAdjointAlgebra(t, a, b) : TenSpcElt, RngIntElt, RngIntElt -> ModMatFld`

Returns the self-adjoint elements of the ab-nucleus of t as a subspace of $\mathrm{End}(U_a)$, with $a, b \in [\![v]\!]$ and $a \neq b$. It is not required that t be in a tensor category with coordinates a and b fused. Unlike other invariants associated to tensors, the self-adjoint algebra is not currently stored with the tensor.

`TensorOverCentroid(t) : TenSpcElt -> TenSpcElt, Hmtp`

If the given tensor t is framed by K-vector spaces, then the returned tensor is framed by E-vector spaces where E is the residue field of the centroid. The returned homotopism is an isotopism of the K-tensors. This only works if the centroid of t is a finite commutative local ring. We employ the algorithms developed by Brooksbank and Wilson [?BW:Module-iso] to efficiently determine if a matrix algebra is cyclic, see Appendix **??**.

Example 3.3. `CentroidUnipotent`

In the context of groups, centroids can be used to recover an underlying field of a matrix group, even if the given group is not input as such. Here we will construct the exponent-p central tensor of the Sylow 2-subgroup of $\mathrm{GL}(3, \mathrm{GF}(2^{10}))$. We will not print the `GrpPC` version of this group as the number of relations is very large.

```
> U := ClassicalSylow(GL(3, 2^10), 2);
> U.3;
[       1    $.1^2         0]
[       0        1         0]
[       0        0         1]
> G := PCPresentation(UnipotentMatrixGroup(U));
> #G eq 2^30;
true
> t := pCentralTensor(G);
> t;
Tensor of valence 3, U2 x U1 >-> U0
U2 : Full Vector space of degree 20 over GF(2)
U1 : Full Vector space of degree 20 over GF(2)
U0 : Full Vector space of degree 10 over GF(2)
```

Even though our tensor right now is $\mathbb{F}_2^{20} \times \mathbb{F}_2^{20} \rightarrowtail \mathbb{F}_2^{10}$, we know it is the 2-dimensional alternating form over $\mathrm{GF}(2^{10})$. We will construct the centroid, and then rewrite our tensor over the centroid to get the tensor we expect.

```
> C := Centroid(t);
> C;
Matrix Algebra of degree 50 and dimension 10 with 1 generator over GF(2)
> IsCyclic(C) and IsSimple(C);
true
> s := TensorOverCentroid(t);
> s;
Tensor of valence 3, U2 x U1 >-> U0
U2 : Full Vector space of degree 2 over GF(2^10)
U1 : Full Vector space of degree 2 over GF(2^10)
U0 : Full Vector space of degree 1 over GF(2^10)
```

Bibliography

Intrinsics