Journal of Algebra 320 (2008) 4020–4029

Contents lists available at ScienceDirect

Journal of Algebra

www.elsevier.com/locate/jalgebra

Testing isomorphism of modules

Peter A. Brooksbank [a,*], Eugene M. Luks [b]

[a] Department of Mathematics, Bucknell University, Lewisburg, PA 17837, USA
[b] Computer and Information Science Department, University of Oregon, Eugene, OR 97403, USA

ARTICLE INFO

Article history:
Received 2 April 2008
Available online 13 August 2008
Communicated by Derek Holt

Keywords:
Matrix algebras
Modules
Polynomial-time algorithm
Weakly-closed set

ABSTRACT

We present a new deterministic algorithm to test constructively for isomorphism between two given finite-dimensional modules of a finitely generated algebra. The algorithm uses only basic field operations; for arbitrary fields, this is not possible with the existing methodology. Furthermore, the number of field operations used by the algorithm is bounded by a polynomial in the length of the input. The algorithm has been implemented in the computer algebra system MAGMA and we report on its performance. Our approach has applications to other problems concerning decompositions of modules.

1. Introduction

In this paper we present algorithms to solve certain fundamental problems in computational representation theory. We are concerned principally with the theoretical complexity of these problems, but we also demonstrate the practicability of the algorithms we present to solve them.

Our algorithms assume an *arithmetic model*, wherein the fundamental steps are basic field operations and computational complexity is determined by counting the *number of these operations*. In particular, an algorithm runs in polynomial time if this number is bounded by a polynomial in the input length. Of course, for finite fields the arithmetical steps run in polynomial time in the usual sense. One advantage of an arithmetic model is its great generality: in principle, our methods apply to arbitrary fields.

Our main result is a new deterministic, polynomial-time algorithm to test for isomorphism between two given finite-dimensional modules. This problem arises naturally in a variety of algorithmic settings, and has been studied extensively.

* Corresponding author.
E-mail addresses: pbrooksb@bucknell.edu (P.A. Brooksbank), luks@cs.uoregon.edu (E.M. Luks).

0021-8693/$ – see front matter
doi:10.1016/j.jalgebra.2008.07.014

In [CIK], Chistov, Ivanyos and Karpinski present a polynomial-time solution to the isomorphism problem over a finite field or a number field. Their algorithm requires the computation of the Jacobson radical of the underlying matrix algebra. However, it is observed in [CIW] that construction of the Jacobson radical over fields of characteristic of p requires computation of pth roots in the base field and, by a result of Frölich and Shepardson [FS], these cannot be computed using basic field operations.

We actually prove the following more general result.

Theorem 3.5. *Given nontrivial modules $\mathcal{M}_1$ and $\mathcal{M}_2$, defined over an arbitrary field, in polynomial time one can find maximal summands of $\mathcal{M}_1$ and $\mathcal{M}_2$ that are isomorphic. Moreover, one can construct $f \in \mathrm{Hom}_\Omega(\mathcal{M}_1, \mathcal{M}_2)$ inducing an isomorphism between these summands.*

The approach we take relies on a classical result that any *weakly-closed subset* (see Section 2.1) of a non-nilpotent matrix algebra contains a non-nilpotent element [Ja, Chapter II]. In Theorem 2.4 we present a constructive proof of this fact. Presented in this generality, the algorithm provides an efficient method to construct non-nilpotent elements in Lie algebras and Jordan algebras. However, the application to module isomorphism requires only the following consequence.

Corollary 2.5. *Given a set X of matrices whose enveloping algebra is not nilpotent, in polynomial time one can find a non-nilpotent element of this algebra as a product of elements of X.*

The ability to construct non-nilpotent elements in this manner has further applications to finding direct sum decompositions of a given module. In particular, we prove the following.

Theorem 3.6. *Given a module $\mathcal{M}$ and nontrivial submodule $\mathcal{T}$, in polynomial time one can find a minimal direct summand of $\mathcal{M}$ containing $\mathcal{T}$.* (*The minimal summand is unique up to isomorphism.*)

Using this result, we also propose a divide-and-conquer approach to computing the socle of a module (Proposition 3.9).

In the case where the defining field is finite and one of the modules is known to be irreducible, the improvement due to Holt and Rees [HR] (see also [IL]) of Parker's *Meat-Axe* algorithm can be adapted to construct an isomorphism between two given modules. Efficient computer implementations of the resulting randomized isomorphism test are distributed with the GAP and MAGMA systems. We conclude the paper by reporting on a MAGMA implementation of our own algorithms.

2. Non-nilpotent matrices

This section is concerned with the construction of non-nilpotent elements in matrix algebras. Our applications rely on a solution to the following:

Problem. Given a set of matrices generating a non-nilpotent algebra, construct a non-nilpotent element of the algebra as a product of elements from the given set.

That such an element exists follows from the results presented in [Ja, Chapter II]. In fact, motivated by Jacobson's treatment we are able more generally to construct a non-nilpotent element in any weakly-closed subset of the given algebra.

In Section 2.1 we introduce the relevant notions and terminology. In Section 2.2 we present a basic version of our algorithm, which we use to establish polynomial timing. Finally, in Section 2.3, we give an alternative version of our algorithm that is more suitable for computer implementation.

2.1. Preliminaries

Let F be a field and let $\mathbf{M}_d(F)$ be the algebra of all $d \times d$ matrices over F. We refer to subalgebras of $\mathbf{M}_d(F)$ as *linear algebras*. An algebra $\mathcal{A}$ is *nilpotent* if $\mathcal{A}^n = 0$ for some positive integer n. It is easy to see that $\mathcal{A} \leqslant \mathbf{M}_d(F)$ is nilpotent if and only if $\mathcal{A}^d = 0$; then $x \in \mathcal{A}$ is *nilpotent* if $x^d = 0$.

For $X \subset \mathbf{M}_d(F)$, denote by $\overline{X}$ the semigroup generated by X. The *enveloping algebra generated by* X is $\mathrm{span}_F(\overline{X})$, the F-linear span of $\overline{X}$: we denote this linear algebra by $\mathrm{Env}(X)$.

Let $V_d(F)$ denote the d-dimensional F-space of row vectors. Then $\mathcal{A} \leqslant \mathbf{M}_d(F)$ acts on $V_d(F)$ by right multiplication. If $W \leqslant V_d(F)$ is invariant under $\mathcal{A}$, then $\mathcal{A}_W$ is the algebra induced by $\mathcal{A}$ on W.

Lemma 2.1. *Given $X \subset \mathbf{M}_d(F)$, and $\mathrm{Env}(X)$-invariant subspace W of $V_d(F)$, in polynomial time one can test whether or not $\mathrm{Env}(X)_W$ is nilpotent.*

Proof. Let $X = \{x_1, \ldots, x_k\}$ and denote by WX the space $Wx_1 + Wx_2 + \cdots + Wx_k$. Initialize $U \leftarrow W$ and proceed as follows:

while $UX \neq U$ **do**
  $U \leftarrow UX$

Clearly $\mathrm{Env}(X)_W$ is nilpotent if and only if $U = 0$ on termination. There are at most d iterations of the loop, and each test $UX = U$ requires $O(|X|d^3)$ field operations. (In case $\mathrm{Env}(X)_W$ is nilpotent, the subspaces obtained in the loop comprise a flag in $V_d(F)$ upon whose quotients $\mathrm{Env}(X)$ acts trivially.) □

Remark 2.2. Strictly speaking, the complexity of the test $UX = U$ above is $O(|X|d^\omega)$, where ω is the exponent of matrix multiplication [CW]. Similar remarks apply to the complexity estimates of other parts of our algorithm. However, since our principal objective is to devise algorithms having polynomial time complexity, we will not concern ourselves with precise statements of this type. Furthermore, this is not a useful practical distinction, since computer algebra systems use variations of classical methods for linear algebra.

Let $\delta : \mathcal{A} \times \mathcal{A} \to F$ be any function. Then $\Delta \subseteq \mathcal{A}$ is *weakly closed with respect to* δ if $xy + \delta(x, y)yx \in \Delta$ for all $x, y \in \Delta$. We say simply that $\Delta \subseteq \mathcal{A}$ is *weakly closed* if the specific δ is understood. The *closure of $X \subseteq \mathcal{A}$ with respect to δ*, denoted $\mathrm{cl}_\delta(X)$, is the smallest subset of $\mathcal{A}$ containing X that is weakly closed with respect to δ. In case δ is identically 0, one has $\mathrm{cl}_\delta(X) = \overline{X}$.

2.2. Basic algorithm

A proof of the following result can be extracted from [Ja, Chapter II].

Theorem 2.3. *Let $\mathcal{A}$ be a linear algebra, and let $X \subseteq \mathcal{A}$ with $\mathrm{Env}(X)$ not nilpotent. Then every weakly-closed subset of $\mathcal{A}$ containing X has a non-nilpotent element.*

In this section we prove a constructive version of this result using the algorithm below. The input is a subset $X = \{x_1, \ldots, x_k\}$ of a linear algebra $\mathcal{A}$ such that $\mathrm{Env}(X)$ is not nilpotent, and a computable function $\delta : \mathcal{A} \times \mathcal{A} \to F$. The output is a non-nilpotent element of $\mathrm{cl}_\delta(X)$. For convenience we define a binary operation $\star_\delta$ on $\mathcal{A}$, where $y \star_\delta z = yz + \delta(y, z)zy$ for all $y, z \in \mathcal{A}$.

```
Algorithm1 (X, δ)
 1  m ← min{i: Env({x_1, ..., x_i}) is not nilpotent}
 2  Y ← {x_1, ..., x_{m−1}}
 3  z ← x_m
 4  while z is nilpotent do
 5    find y ∈ Y such that y ⋆_δ z ∉ Env(Y)
 6    if Env(Y ∪ {y ⋆_δ z}) is nilpotent then
 7      Y ← Y ∪ {y ⋆_δ z}
 8    else
 9      z ← y ⋆_δ z
10  return z
```

Theorem 2.4. *Given $X \subseteq \mathbf{M}_d(F)$ such that* $\mathrm{Env}(X)$ *is not nilpotent, and any computable function δ,* `Algorithm1` *returns a non-nilpotent element of* $\mathrm{cl}_\delta(X)$. *If δ is computable in polynomial time, then the algorithm runs in polynomial time.*

Proof. First note that the following are all invariants of the **while** loop beginning on line 4: $\mathrm{Env}(Y)$ is nilpotent; $\mathrm{Env}(Y \cup \{z\})$ is not nilpotent; and $z \in \mathrm{cl}_\delta(X)$. In view of the latter, it suffices to show that the loop terminates in polynomial time.

Consider a fixed iteration of the loop (in which z is assumed nilpotent). Let

$$A_i = \mathrm{span}_F\left(\bigcup_{j+k \geqslant i} z^j Y^k\right) \subset \mathbf{M}_d(F) \quad \text{for positive integers } i.$$

Since $A_1 \neq 0$ and $A_{2d} = 0$ (the latter since z and $\mathrm{Env}(Y)$ are nilpotent),

$$A_1 \geqslant A_2 \geqslant A_3 \geqslant \cdots \geqslant A_{2d-1} \geqslant A_{2d} = 0$$

is a nontrivial flag in the F-space $\mathbf{M}_d(F)$. Clearly $A_i Y \subseteq A_{i+1}$ so $\mathrm{Env}(Y)$ acts (by right multiplication) trivially on the quotients of the flag.

The successful execution of the iteration depends only on the existence of $y \in Y$ such that $y \star_\delta z \notin \mathrm{Env}(X)$ (line 5). Suppose, to the contrary, that $Y \star_\delta z \subseteq \mathrm{Env}(Y)$, so that

$$Yz \subseteq \mathrm{span}_F(zY) + \mathrm{Env}(Y).$$

It follows that $A_i z \subseteq A_i$, and hence that $\mathrm{Env}(Y \cup \{z\})$ acts on the flag. Since z is nilpotent, it acts trivially on the quotients of some refinement of the flag. Then, however, $\mathrm{Env}(Y \cup \{z\})$ acts trivially on those quotients, which is absurd since $\mathrm{Env}(Y \cup \{z\})$ is not nilpotent.

It remains only to establish polynomial timing. First note that the loop terminates after at most $\binom{d}{2}$ iterations. For, adding $y \star_\delta z$ to Y (line 7) increases the dimension of $\mathrm{Env}(Y)$, which, since $\mathrm{Env}(Y)$ is nilpotent, can occur at most $\binom{d-1}{2}$ times, while z can be reassigned to $y \star_\delta z$ (line 9) at most $d-1$ times for the same reason.

Finally, each iteration requires a polynomial number of field operations: up to $|Y|$ elements $y \star_\delta z$ are constructed in line 5, each in polynomial time; the test $y \star_\delta z \notin \mathrm{Env}(Y)$ is a membership test in an F-space of dimension less than d^2; and the nilpotence or otherwise of $\mathrm{Env}(Y \cup \{y \star_\delta z\})$ is decided using Lemma 2.1. □

The following consequence of Theorem 2.4 is central to our applications.

Corollary 2.5. *Given $X \subset \mathbf{M}_d(F)$ such that* $\mathrm{Env}(X)$ *is not nilpotent, in polynomial time one can find $z \in \overline{X}$ not nilpotent and write $z = xz'$ with $x \in X$ and $z' \in \overline{X} \cup \{1\}$.*

Proof. Apply Theorem 2.4 to X, taking δ to be the trivial function, to construct a suitable $z \in \overline{X}$. Recording how z is constructed from X in `Algorithm1`, one readily obtains a suitable factorization. □

2.3. Practical alternative

We now present an alternative version of the algorithm that is more suitable for computer implementation. Indeed our own MAGMA implementation, which we discuss in Section 4, contains a function based on the following description.

```
Algorithm2 (X, δ)
 1   m ← min{i: Env({x_1, ..., x_{i−1}}) is not nilpotent}
 2   Y ← {x_1, ..., x_{m−1}}
 3   z ← x_m
 4   W ← V_d(F)
 5   while z is nilpotent do
 6     while dim(WY + Wz) < dim W do
 7       W ← WY + Wz
 8     find y ∈ Y such that W(y ⋆_δ z) ⊄ WY
 9     if Env(Y ∪ {y ⋆_δ z})_W is nilpotent then
10       Y ← Y ∪ {y ⋆_δ z}
11     else
12       z ← y ⋆_δ z
13   return z
```

Theorem 2.6. *Given $X \subseteq \mathbf{M}_d(F)$ such that* $\mathrm{Env}(X)$ *is not nilpotent, and any function δ,* `Algorithm2` *returns a non-nilpotent element of* $\mathrm{cl}_\delta(X)$. *If δ is computable in polynomial time, then the algorithm runs in polynomial time.*

Proof. The outer loop beginning on line 5 has the following invariants: W is $Y \cup \{z\}$-invariant; $\mathrm{Env}(Y)_W$ is nilpotent; and $\mathrm{Env}(Y \cup \{z\})_W$ is not nilpotent. In view of the latter, the inner loop beginning on line 6 terminates with $W \neq 0$ and $W = WY + Wz$.

Once again we must establish the existence of a suitable $y \in Y$ in line 8. Suppose, to the contrary, that $W(Y \star_\delta z) \subseteq WY$. Since W is invariant under $Y \cup \{z\}$, we have $WYz \subseteq WY$. But then $Wz = (WY + Wz)z \subseteq WY + Wz^2$ and, inductively, $W \subseteq WY + Wz^i$ for all integers $i \geqslant 1$. Since z is nilpotent, we have $W \subseteq WY$, contradicting the nilpotence of $\mathrm{Env}(Y)_W$.

To establish polynomial timing, note that W can be reassigned at most $d-1$ times (line 7). If Y is reassigned to $Y \cup \{y \star_\delta z\}$ (line 10), then the dimension of WY increases; this can occur at most $\dim(W)$ times for fixed W. The value of z can be reassigned to $y \star_\delta z$ (line 12) at most $d-1$ times, as in the basic version. □

Remark 2.7. Although both `Algorithm1` and `Algorithm2` perform $O(d^2)$ iterations in the worst case, the tests performed within each iteration of the former are more expensive than their counterparts in the latter. For example, `Algorithm1` tests membership in $\mathrm{Env}(Y)$, of dimension $O(d^2)$, whereas `Algorithm2` tests membership in a space of dimension at most d.

3. Applications to modules

Throughout this section Ω will denote a finitely generated algebra over an arbitrary field F. An Ω-module $\mathcal{M}$ is specified by the action of a fixed set $\{\omega_1, \ldots, \omega_n\}$ of generators for Ω on some finite-dimensional vector space over F. Thus $\mathcal{M}$ is input via $\{a_1, \ldots, a_n\} \subset \mathbf{M}_d(F)$, where $d = \dim_F(\mathcal{M})$.

A fundamental component of the algorithms in this section is the construction of the space of all homomorphisms between two given modules. For given Ω-modules $\mathcal{M}_1$ and $\mathcal{M}_2$, of dimensions d_1 and d_2 respectively, we regard $\mathrm{Hom}_\Omega(\mathcal{M}_1, \mathcal{M}_2)$ as a subspace of $\mathbf{M}_{d_1 \times d_2}(F)$, the space of all $d_1 \times d_2$ matrices with entries in F. In this way, $\mathrm{Hom}_\Omega(\mathcal{M}_1, \mathcal{M}_2)$ is easily expressed as the solution space of a system of linear equations. In particular, using standard linear-algebraic techniques we have the following.

Lemma 3.1. *Given Ω-modules $\mathcal{M}_1$ and $\mathcal{M}_2$, in polynomial time one can compute an F-basis for* $\mathrm{Hom}_\Omega(\mathcal{M}_1, \mathcal{M}_2)$.

Remark 3.2. Although the construction of $\mathrm{Hom}_\Omega(\mathcal{M}_1, \mathcal{M}_2)$ is elementary from a complexity viewpoint, it is a substantial bottleneck in practical implementations. We will discuss this issue further in Section 4.

3.1. Splitters

Key to our various applications is the ability to construct isomorphisms between direct summands of two given Ω-modules if such exist. If $\mathcal{N}_1$ is a submodule of $\mathcal{M}_1$ and $f \in \mathrm{Hom}_\Omega(\mathcal{M}_1, \mathcal{M}_2)$, then we denote by $\mathcal{N}_1 f$ the f-image of $\mathcal{N}_1$ in $\mathcal{M}_2$. We say that $\mathcal{M}_1 = \mathcal{N}_1 \oplus \mathcal{K}_1$ is an *f-decomposition* if $\mathcal{N}_1 \neq 0$, $\ker(f) \leqslant \mathcal{K}_1$, and $\mathcal{N}_1 f$ a direct summand of $\mathcal{M}_2$. If $\mathcal{M}_1$ has an f-decomposition then we say that f is a *splitter*.

Lemma 3.3. *Let $f \in \mathrm{Hom}_\Omega(\mathcal{M}_1, \mathcal{M}_2)$. Then f is a splitter if and only if there exists $g \in \mathrm{Hom}_\Omega(\mathcal{M}_2, \mathcal{M}_1)$ such that $fg \in \mathrm{End}_\Omega(\mathcal{M}_1)$ is not nilpotent.*

Proof. First suppose that f is a splitter and write $\mathcal{M}_i = \mathcal{N}_i \oplus \mathcal{K}_i$ $(i = 1, 2)$, where $0 \neq \mathcal{N}_1 \cong \mathcal{N}_1 f = \mathcal{N}_2$. If $\alpha : \mathcal{N}_1 \to \mathcal{N}_2$ is the isomorphism induced by f, let g be the element of $\mathrm{Hom}_\Omega(\mathcal{M}_2, \mathcal{M}_1)$ inducing α^{-1} on $\mathcal{N}_2$ and 0 on $\mathcal{K}_2$. Then $fg \in \mathrm{End}_\Omega(\mathcal{M}_1)$ is the identity on $\mathcal{N}_1$ and hence is not nilpotent.

Conversely, let g be any element of $\mathrm{Hom}_\Omega(\mathcal{M}_2, \mathcal{M}_1)$ such that fg is not nilpotent. Let $s = fg$ and $t = gf$. Then $\mathcal{M}_1 = \mathcal{M}_1 s^d \oplus \ker(s^d)$ and $\mathcal{M}_2 = \mathcal{M}_2 t^d \oplus \ker(t^d)$ are Ω-module decompositions of $\mathcal{M}_1$ and $\mathcal{M}_2$, where $d = \max\{\dim(\mathcal{M}_1), \dim(\mathcal{M}_2)\}$. Furthermore $\mathcal{M}_1 s^d \neq 0$, $(\mathcal{M}_1 s^d) f = \mathcal{M}_2 t^d$ and $\ker(f) \leqslant \ker(s^d)$. □

This result provides the basis of an elementary test for whether a given homomorphism is a splitter.

Lemma 3.4. *Given Ω-modules $\mathcal{M}_1$ and $\mathcal{M}_2$, and $f \in \mathrm{Hom}_\Omega(\mathcal{M}_1, \mathcal{M}_2)$, in polynomial time one can decide whether or not f is a splitter. If it is, moreover, one can find an f-decomposition of $\mathcal{M}_1$.*

Proof. Use Lemma 3.1 to compute a basis, C, for $\mathrm{Hom}_\Omega(\mathcal{M}_2, \mathcal{M}_1)$. Now use Lemma 2.1 to decide whether or not $\mathrm{Env}(fC) \leqslant \mathrm{End}_\Omega(\mathcal{M}_1)$ is nilpotent.

Observe that $f \cdot \mathrm{Hom}_\Omega(\mathcal{M}_2, \mathcal{M}_1) = \mathrm{span}_F(fC) = \mathrm{Env}(fC)$. Thus, if $\mathrm{Env}(fC)$ is nilpotent, by Lemma 3.3, f is not a splitter.

On the other hand, if $\mathrm{Env}(fC)$ is not nilpotent, use Corollary 2.5 to construct a non-nilpotent $s \in \overline{fC}$, so that $s = fg$ for some $g \in \mathrm{Hom}_\Omega(\mathcal{M}_2, \mathcal{M}_1)$. Hence, by Lemma 3.3 and its proof, $\mathcal{M}_1 = \mathcal{M}_1 s^d \oplus \ker(s^d)$ is an f-decomposition of $\mathcal{M}_1$. □

3.2. Testing isomorphism

Our first application of Lemma 3.4 is a polynomial-time, constructive test for isomorphism between Ω-modules. In fact, we prove the following stronger result.

Theorem 3.5. *Given nontrivial Ω-modules $\mathcal{M}_1$ and $\mathcal{M}_2$, in polynomial time one can find maximal summands of $\mathcal{M}_1$ and $\mathcal{M}_2$ that are isomorphic. Moreover, one can construct $f \in \mathrm{Hom}_\Omega(\mathcal{M}_1, \mathcal{M}_2)$ inducing an isomorphism between these summands.*

Proof. The existence of the isomorphic summands is a consequence of the Krull–Schmidt theorem for modules [AF, Theorem 12.9], which also establishes the uniqueness of such summands up to isomorphism. Initialize $\mathcal{M}_1^* \leftarrow 0$, $f \leftarrow 0$, $\mathcal{L}_1 \leftarrow \mathcal{M}_1$, $\mathcal{L}_2 \leftarrow \mathcal{M}_2$, $B \leftarrow$ basis for $\mathrm{Hom}_\Omega(\mathcal{L}_1, \mathcal{L}_2)$, and proceed as follows.

```
1  while ∃ a splitter b ∈ B do
2    write L1 = N1 ⊕ K1 and L2 = N1 b ⊕ K2
3    M1* ← M1* ⊕ N1 and f ← f⊕ restriction of b to N1
4    L1 ← K1 and L2 ← K2
5    B ← basis for Hom_Ω(L1, L2)
6  return f, M1*.
```

Note that the control of the loop is tested in line 1 using Lemma 3.4, which also produces the decompositions of $\mathcal{L}_1$ and $\mathcal{L}_2$ in line 2.

It is clear that $\mathcal{M}_1^*$ is a summand of $\mathcal{M}_1$, and that $\mathcal{M}_1^* f$ is a summand of $\mathcal{M}_2$ isomorphic to $\mathcal{M}_1^*$. It remains to show that $\mathcal{M}_1^*$ is maximal.

First note that $\mathcal{M}_1 = \mathcal{M}_1^* \oplus \mathcal{L}_1$ is an invariant of the loop. Suppose, at the start of some iteration of the loop, that $\mathcal{L}_1$ has a nontrivial summand, $\mathcal{S}_1$, isomorphic to a summand, $\mathcal{S}_2$, of $\mathcal{L}_2$. It suffices to show that any basis for $\mathrm{Hom}_\Omega(\mathcal{L}_1, \mathcal{L}_2)$ contains a splitter.

Let B^* be any basis for $\mathrm{Hom}_\Omega(\mathcal{L}_1, \mathcal{L}_2)$ and C^* any basis for $\mathrm{Hom}_\Omega(\mathcal{L}_2, \mathcal{L}_1)$. Let s be an element of $\mathrm{Hom}_\Omega(\mathcal{L}_1, \mathcal{L}_2)$ inducing an isomorphism $\mathcal{S}_1 \to \mathcal{S}_2$, and let t be an element of $\mathrm{Hom}_\Omega(\mathcal{L}_2, \mathcal{L}_1)$ inducing an isomorphism $\mathcal{S}_2 \to \mathcal{S}_1$. Then

$$st \in \mathrm{Hom}_\Omega(\mathcal{L}_1, \mathcal{L}_2) \cdot \mathrm{Hom}_\Omega(\mathcal{L}_2, \mathcal{L}_1) = \mathrm{span}_F(B^*) \cdot \mathrm{span}_F(C^*) \leqslant \mathrm{Env}(B^* C^*)$$

is nonsingular on $\mathcal{S}_1$, and hence is not nilpotent. Therefore, by Corollary 2.5, there exists $z = (b^* c^*) z'$ not nilpotent, with $b^* \in B^*$, $c^* \in C^*$ and $z' \in \overline{B^* C^*} \cup \{1\}$. Hence $z = b^* g$ for some $g \in \mathrm{Hom}_\Omega(\mathcal{L}_2, \mathcal{L}_1)$, whence b^* is a splitter by Lemma 3.3. □

3.3. Decomposing modules

The problem of computing a nontrivial direct sum decomposition of a given module (or establishing that the module is indecomposable) is considered, for example, in [CIK]. There, a polynomial-time Las Vegas algorithm is given that solves this problem for finite fields and certain algebraic number fields.

As a further application of Proposition 3.4, we present a deterministic algorithm (which again applies to arbitrary fields) to solve a related decomposition problem.

Theorem 3.6. *Given an Ω-module $\mathcal{M}$ and nontrivial submodule $\mathcal{T}$, in polynomial time one can find a minimal direct summand of $\mathcal{M}$ containing $\mathcal{T}$.*

Proof. Let $\pi : \mathcal{M} \to \mathcal{M}/\mathcal{T}$ be the natural map. If $\mathcal{M} = \mathcal{N} \oplus \mathcal{K}$ with $\mathcal{N} \neq 0$ and $\mathcal{T} = \ker(\pi) \leqslant \mathcal{K}$, we have $\mathcal{M}/\mathcal{T} = \mathcal{N}\pi \oplus (\mathcal{K}/\mathcal{T})$, so that $\pi \in \mathrm{Hom}_\Omega(\mathcal{M}, \mathcal{M}/\mathcal{T})$ is a splitter. Thus one may obtain *some* proper summand $\mathcal{K}$ containing $\mathcal{T}$ by applying Lemma 3.4 to π. A minimal such summand is obtained by iteration. □

The minimal summand constructed in the theorem need not be unique, as one can see with the $\mathbb{Z}$-module $\mathcal{M} = \mathbb{Z}_4 \oplus \mathbb{Z}_2$. Here, $(1,0)$ and $(1,1)$ generate distinct subgroups of order 4 (and hence distinct direct summands of $\mathcal{M}$) that both contain the submodule generated by $(2,0)$, which has no complement. However, we note the following.

Theorem 3.7. *Let $\mathcal{M}$ be an Ω-module and let $\mathcal{T}$ be a nontrivial submodule. If $\mathcal{K}_1$ and $\mathcal{K}_2$ are summands of $\mathcal{M}$, both minimal with respect to containing $\mathcal{T}$, then there is an automorphism of $\mathcal{M}$ sending $\mathcal{K}_1$ to $\mathcal{K}_2$.*

Proof. Suppose, for $i = 1, 2$, that $\mathcal{M} = \mathcal{N}_i \oplus \mathcal{K}_i$, where $\mathcal{K}_i$ is minimal with respect to containing $\mathcal{T}$. Let $f_i \in \mathrm{Hom}_\Omega(\mathcal{M}, \mathcal{K}_{3-i})$ be projection onto $\mathcal{K}_{3-i}$, and let $g_i \in \mathrm{Hom}_\Omega(\mathcal{K}_i, \mathcal{K}_{3-i})$ be the restriction of f_i to $\mathcal{K}_i$. Then g_i is the identity on $\mathcal{T}$ so that $g_i g_{3-i} \in \mathrm{End}_\Omega(\mathcal{K}_i)$ is not nilpotent. Hence $\mathcal{K}_i (g_i g_{3-i})^{\dim \mathcal{M}}$ is a direct summand of $\mathcal{K}_i$ containing $\mathcal{T}$. By minimality $\mathcal{K}_i = \mathcal{K}_i (g_i g_{3-i})^{\dim \mathcal{M}}$, whence $g_1 : \mathcal{K}_1 \to \mathcal{K}_2$ is an isomorphism. It follows that $\mathcal{N}_1$ and $\mathcal{N}_2$ are also isomorphic, and g_1 can be extended to an automorphism of $\mathcal{M}$. □

Remark 3.8. Using the same ideas one can derive a variation of this result for groups. For a finite group G and subgroup A, let

$$\mathfrak{S}_{G,A} = \{S\colon\ A \leqslant S \leqslant G \text{ and } G = N \rtimes S \text{ with } N \text{ a normal subgroup of } G\}.$$

If S_1 and S_2 are minimal elements of $\mathfrak{S}_{G,A}$, then $S_1 \cong S_2$ (though this isomorphism does not necessarily extend to an automorphism of G).

As one application of Theorem 3.6 we propose a divide-and-conquer approach to computing the socle of a module.

Proposition 3.9. *Let $\mathcal{M}$ be an Ω-module and let $\mathcal{T}$ be any proper submodule. Given* $\mathrm{Soc}(\mathcal{T})$ *and* $\mathrm{Soc}(\mathcal{M}/\mathrm{Soc}(\mathcal{T}))$*, in polynomial time one can compute* $\mathrm{Soc}(\mathcal{M})$.

Proof. Let $\mathcal{S}_0 = \mathrm{Soc}(\mathcal{T})$ and $\mathcal{S}_1/\mathcal{S}_0 = \mathrm{Soc}(\mathcal{M}/\mathcal{S}_0)$, so that $\mathcal{S}_0 \leqslant \mathrm{Soc}(\mathcal{M}) \leqslant \mathcal{S}_1$. Use Lemma 3.6 to write $\mathcal{S}_1 = \mathcal{N} \oplus \mathcal{K}$, where $\mathcal{K}$ is minimal with respect to containing $\mathcal{S}_0$. We claim that $\mathcal{S}_0 \oplus \mathcal{N} = \mathrm{Soc}(\mathcal{M})$.

Let $\pi : \mathcal{S}_1 \to \mathcal{S}_1/\mathcal{S}_0$ be the natural map, so that $\mathcal{S}_1/\mathcal{S}_0 = (\mathcal{N}\pi) \oplus (\mathcal{K}/\mathcal{S}_0)$. Since $\mathcal{S}_1/\mathcal{S}_0$ is semisimple, $\mathcal{N}\pi \cong \mathcal{N}$ is semisimple. Hence

$$\mathrm{Soc}(\mathcal{M}) = \mathrm{Soc}(\mathcal{S}_1) = \mathrm{Soc}(\mathcal{N}) \oplus \mathrm{Soc}(\mathcal{K}) = \mathcal{N} \oplus \mathrm{Soc}(\mathcal{K}).$$

It suffices to show that $\mathcal{S}_0 = \mathrm{Soc}(\mathcal{K})$. Suppose, to the contrary, that $\mathcal{S}_0$ is a proper summand of $\mathrm{Soc}(\mathcal{K})$, say $\mathrm{Soc}(\mathcal{K}) = \mathcal{S}_0 \oplus \mathcal{N}^*$. Since $\mathcal{K}/\mathcal{S}_0$ is semisimple, there exists a proper submodule $\mathcal{K}^*$ of $\mathcal{K}$, containing $\mathcal{S}_0$, such that $\mathcal{K}/\mathcal{S}_0 = (\mathcal{K}^*/\mathcal{S}_0) \oplus (\mathrm{Soc}(\mathcal{K})/\mathcal{S}_0)$. Hence $\mathcal{K} = \mathcal{K}^* + \mathrm{Soc}(\mathcal{K}) = \mathcal{K}^* + (\mathcal{S}_0 \oplus \mathcal{N}^*) = \mathcal{K}^* \oplus \mathcal{N}^*$, with $\mathcal{S}_0 \leqslant \mathcal{K}^*$, contradicting the minimality of $\mathcal{K}$. □

This yields a Las Vegas algorithm to compute the socle of a module defined over a finite field (see [LW] for another approach to this problem).

Theorem 3.10. *Let $\mathcal{M}$ be an Ω-module defined over a finite field F. Then* $\mathrm{Soc}(\mathcal{M})$ *can be computed in polynomial time using a Las Vegas algorithm.*

Proof. Use [Ró] to test whether $\mathcal{M}$ is irreducible (this test is Las Vegas polynomial time for large finite fields). If it is, then $\mathcal{M} = \mathrm{Soc}(\mathcal{M})$ and we are done. If it is not, then [Ró] yields a proper submodule $\mathcal{T}$. Recursively compute $\mathrm{Soc}(\mathcal{T})$ and $\mathrm{Soc}(\mathcal{M}/\mathrm{Soc}(\mathcal{T}))$, and then use Proposition 3.9 to construct $\mathrm{Soc}(\mathcal{M})$. □

Remark 3.11. For a practical version of Theorem 3.10, one should instead use the algorithm of Holt and Rees [HR] to test for irreducibility. Their algorithm is also Las Vegas, but has better asymptotic complexity than Rónyai's algorithm; in addition there are several highly effective implementations of their algorithm available.

4. Implementation

The various algorithms presented in Sections 2 and 3 have been implemented in Magma [BCP] and are publicly available. In this section we briefly discuss some practical issues pertaining to the implementation.

4.1. Computing $\mathrm{Hom}_\Omega(\mathcal{M}_1, \mathcal{M}_2)$

This appears to be the most formidable obstacle from a practical viewpoint.

Let $d_i = \dim_F(\mathcal{M}_i)$ for $i = 1, 2$, and suppose that $\mathcal{M}_i$ is generated by sets of n matrices from $\mathbf{M}_{d_i}(F)$. Then we seek the solution to a linear system of size $O(nd_1d_2)$ in d_1d_2 unknowns. If $d_1 \leqslant d_2 = d$, computation of the solution space of this system costs $O(nd^6)$ field operations (cf. Remark 2.2), which is prohibitive even for moderate values of d. Furthermore the space required to store elements of an arbitrary field may increase rapidly throughout such a computation.

In the case of finite fields there are efficient methods to handle such computations. The Magma function `AHom`, implemented by Steel, is very effective and our implementation uses it wherever

Table 1
`SummandIsomorphism` versus `IsIsomorphic` for modules over GF(2)

b	m	`ModuleIso`	`IsIsomorphic`
24	2	0.042	0.028
16	3	0.046	0.028
12	4	0.055	0.031
8	6	0.073	0.038
4	12	0.095	0.093
3	16	0.120	0.490
2	24	0.144	832.592

possible. The methods underlying this function were developed by Leedham-Green and Steel, but remain unpublished. An alternative approach to the problem by Lux and Szöke is described in [LS]. Steel recently extended the functionality of `AHom` so that it can now be applied to modules defined over the rationals.

4.2. Testing isomorphism

We now describe a series of tests carried out with our implementation of the algorithm to construct an isomorphism between summands of two modules. We henceforth refer to this implementation via its function name `SummandIsomorphism`.

Irreducible modules over finite fields. As noted in the introduction, if one of the input isomorphic modules is known to be irreducible, then an isomorphism can be readily constructed using standard Meat-Axe machinery [HR]. Although we did not anticipate that `SummandIsomorphism` would be competitive with the MAGMA default function `IsIsomorphic` in this special case, we do not come away too badly even here.

We compared the two functions by constructing invariant forms for symplectic groups $\mathrm{Sp}(d,q)$ for various values of d and q. This is equivalent to computing an isomorphism between the natural module $V = V_d(\mathrm{GF}(q))$ for G and its dual module V^*. To define V we took matrix generators for a random conjugate of the standard MAGMA copy of $\mathrm{Sp}(d,q)$; to define V^* we took the inverse-transposes of these generators. In a variety of tests conducted with symplectic groups of degree up to 1000 defined over fields of moderate size, `IsIsomorphic` constructed the desired form roughly twice as quickly as `SummandIsomorphism`. (This is to be expected, since `SummandIsomorphism` essentially computes *two* spaces of homomorphisms rather than one.)

General modules over finite fields. There is a randomized method to construct an isomorphism $\mathcal{M}_1 \to \mathcal{M}_2$ from a basis for $\mathrm{Hom}_\Omega(\mathcal{M}_1, \mathcal{M}_2)$. Namely, construct random elements of $\mathrm{Hom}_\Omega(\mathcal{M}_1, \mathcal{M}_2)$ until a nonsingular element is found.

This naïve approach performs badly in situations where the proportion of invertible elements in $\mathrm{Hom}_\Omega(\mathcal{M}_1, \mathcal{M}_2)$ is small. To construct instances of such module pairs, one may proceed as follows.

1. Fix positive integers b and m and a finite field $F = \mathrm{GF}(q)$.
2. Write out a set of matrices, X, generating the simple algebra $\mathbf{M}_b(F)$.
3. Set $Y \leftarrow \{\mathrm{diag}(x, x, \ldots, x)\colon\ x \in X\} \subset \mathbf{M}_{bm}(F)$.
4. Define $\mathcal{M}_1$ using the set Y, and define $\mathcal{M}_2$ using a random conjugate of Y.

Then $\mathrm{End}_\Omega(\mathcal{M}_1) \cong \mathrm{GF}(q)^m$, so the probability that an element of $\mathrm{Hom}_\Omega(\mathcal{M}_1, \mathcal{M}_2)$ is nonsingular is $(1 - 1/q)^m$.

Using this construction, we fixed $d = 48$ and $q = 2$ and compared the performance of `SummandIsomorphism` with that of `IsIsomorphic` for various choices of input parameters (b, m) with $bm = 48$. Using an Intel Xeon computer with two 2.2 GHz processors and 2GB main memory, the average of 20 runs for each implementation was taken and recorded in Table 1. As expected, the results indicate that our implementation performs comparatively better as m increases.

Naturally, there are many other performance comparisons that one could run (we have, for example, conducted tests with a variety of permutation modules). However, we do not expect that our

implementation will perform significantly better than the MAGMA default for arbitrary modules over finite fields.

Modules over infinite fields. We have used `SummandIsomorphism` to construct isomorphisms between modules defined over the rationals, and also between modules defined over infinite fields of positive characteristic. In the former case, the `AHom` function in MAGMA works very well, and we have successfully tested isomorphism between rational modules in dimension up to 100 in reasonable time. For other infinite fields $\mathrm{Hom}_{\Omega}(\mathcal{M}_1, \mathcal{M}_2)$ is presently obtained as the solution to a linear system by brute force. In order for this to be practical one must carefully manage linear algebraic computations in order to avoid integer explosions. We have thus far succeeded with such examples only in very modest dimensions.

Acknowledgments

The authors benefited from useful correspondence with G. Ivanyos, L. Rónyai and A. Steel.

References

[AF] F.W. Anderson, K.R. Fuller, Rings and Categories of Modules, Springer-Verlag, 1992.
[BCP] W. Bosma, J. Cannon, C. Playoust, The MAGMA Algebra System I: The user language, J. Symbolic Comput. 24 (1997) 235–265.
[CIK] A. Chistov, G. Ivanyos, M. Karpinski, Polynomial-time algorithms for modules over finite dimensional algebras, in: Proceedings Int. Symp. on Symbolic and Algebraic Computation (ISSAC), 1997, pp. 68–74.
[CIW] A. Cohen, G. Ivanyos, D. Wales, Finding the radical of an algebra of linear transformations, J. Pure Appl. Algebra 117/118 (1997) 177–193.
[CW] D. Coppersmith, S. Winograd, Matrix multiplication via arithmetic progressions, J. Symbolic Comput. 9 (1990) 251–280.
[FS] A. Frölich, J.C. Shepardson, Effective procedures in field theory, R. Soc. Lond. Philos. Trans. A 248 (1955–1956) 407–432.
[HR] D.F. Holt, S. Rees, Testing modules for irreducibility, J. Austral. Math. Soc. Ser. A 57 (1994) 1–16.
[Ja] N. Jacobson, Lie Algebras, Interscience Tracts in Pure and Applied Mathematics, Interscience Publishers, 1962.
[IL] G. Ivanyos, K. Lux, Treating the exceptional cases of the Meat-axe, Experiment. Math. 9 (3) (2000) 373–381.
[LS] K. Lux, M. Szöke, Homomorphism spaces between modules over finite dimensional algebras, Experiment. Math. 12 (2003) 91–98.
[LW] K. Lux, M. Wiegelmann, Determination of socle series using the condensation method, in: Computational Algebra and Number Theory, Milwaukee, WI, 1996, J. Symbolic Comput. 31 (2001) 163–178.
[Ró] L. Rónyai, Computing the structure of finite algebras, J. Symbolic Comput. 9 (1990) 355–373.