

## MATCHING SIMPLE MODULES OF CONDENSED ALGEBRAS

FELIX NOESKE

*Abstract*

Let  $A$  be a finite dimensional algebra over a finite field  $F$ . Condensing an  $A$ -module  $V$  with two different idempotents  $e$  and  $e'$  leads to the problem that to compare the composition series of  $Ve$  and  $Ve'$ , we need to match the composition factors of both modules. In other words, given a composition factor  $S$  of  $Ve$ , we have to find a composition factor  $S'$  of  $Ve'$  such that there exists a composition factor  $\hat{S}$  of  $V$  with  $\hat{S}e \cong S$  and  $\hat{S}e' \cong S'$ , or prove that no such  $S'$  exists. In this note, we present a computationally tractable solution to this problem.

1. *Introduction*

In computational representation theory, condensation has become an indispensable tool. In particular the Modular Atlas project [1] whose objective it is to classify the irreducible modular representations of the almost simple groups featured in the Atlas [2], owes its continuous progress to this method: The current open questions involving the sporadic simple groups pose challenging problems for current generation's computers, as the size of the representing matrices impedes a direct application of the Meat-Axe (see [13] for an introduction).

To be precise, let  $A$  be a finite dimensional algebra over some finite field  $F$  of characteristic  $p$ . Then given a finite dimensional  $A$ -module  $V$ , *condensation* refers to the application of the restriction functor (also called condensation functor) from  $\text{mod-}A$  to  $\text{mod-}eAe$ . In other words, instead of considering the whole, computationally intractable module  $V$ , we consider the much smaller dimensional subspace  $Ve$  as a module for the unital algebra  $eAe$ .

The condensation functor possesses several useful properties (confer, for example, [3, 14]), among which the preservation of simple modules is of great importance: If  $V$  is a simple  $A$ -module, then  $Ve$  is either 0 or a simple  $eAe$ -module, and all simple  $eAe$ -modules arise in this manner. In fact, if all simple  $A$ -modules condense to nonzero  $eAe$ -modules, then condensation establishes a Morita-equivalence between the two algebras  $A$  and  $eAe$ .

In practice, owing to the size of the  $A$ -module  $V$ , a paradigm of condensation is to avoid any explicit computation in this huge space. Therefore the availability of idempotents yielding sufficiently small-dimensional condensed modules and also a Morita-equivalence of algebras is limited: We need to restrict ourselves to idempotents for which we have access to algorithms adhering to the paradigm.

---

Supported by the DFG grant HI 895/1-1

Received 14 September 2007, revised 22 January 2008; *published* 1 August 2008.

2000 Mathematics Subject Classification 20C40

© 2008, Felix Noeske

Currently, there are many available algorithms to efficiently condense a vast array of modules for a group algebra  $FG$ , if the condensation idempotent  $e$  is chosen to be a so-called *linear* idempotent, i.e.,  $e$  is a central primitive idempotent corresponding to a linear character  $\lambda$  of some subgroup  $K \leq G$  whose order is coprime to  $p$  (see [5, 8, 10, 15] where  $\lambda$  is trivial, see [7, 11] where  $\lambda$  may be arbitrary). The common theme of all available condensation algorithms, and the source of their efficacy, is a special choice of basis which allows us to compute the action of a condensed group element  $ege$  for some  $g \in G$  on the condensed module  $Ve$  without any explicit calculation in the whole of  $V$ . In particular, we may generally distinguish two parts in every implementation of condensation: The first computes the images of the basis elements of  $Ve$  under the action of  $eg$ , the second projects these images onto  $Ve$  by applying the projection induced by  $e$ .

Unfortunately, choosing the subgroup  $K$  too small, invariably yields a condensed module whose size may still not let the envisaged calculation fit within the bounds of the available computational resources. Thus in practice, a linear idempotent which condenses some simple  $A$ -modules to zero is often our only choice.

When determining composition series of modules via condensation, any simple  $A$ -module which is annihilated by the condensation idempotent  $e$  is a blind spot: The information provided by the condensed module is incomplete. To gain the full knowledge of a composition series, we therefore condense with additional idempotents, until we are sure to cover all simple  $A$ -modules, i.e., no simple  $A$ -module condenses to the zero module under all condensation idempotents chosen.

We then face a new problem: Given two idempotents  $e, e' \in A$ , a composition factor  $S$  of  $Ve$ , and a composition factor  $S'$  of  $Ve'$ , determine if there exists a composition factor  $\hat{S}$  of  $V$  such that  $\hat{S}e \cong S$  and  $\hat{S}e' \cong S'$ . If such an  $A$ -module exists, we say that we can *match*  $S$  to  $S'$ , and that  $\hat{S}$  is an *origin* of  $S$  and  $S'$ .

In this note we present a new method for the matching of simple modules for two condensed algebras. Instead of matching  $S$  and  $S'$  by attempting the usually hopeless task of determining their origins in  $\mathbf{mod}\text{-}A$ , and testing these for isomorphism, our approach ‘zig-zags’ between both algebras  $eAe$  and  $e'Ae'$  via their common ambient algebra  $A$ . By taking care that all major computations occur within condensed modules, we can adhere to the condensation paradigm formulated above. However, as the algebra  $A$  provides the link between  $eAe$  and  $e'Ae'$ , we cannot avoid computing within  $\mathbf{mod}\text{-}A$  entirely. But we have taken care to reduce these computations to a minimum, and illustrate in Sections 4 and 5 that they do not infringe upon the method’s practical applicability.

This paper is structured as follows: We begin in Section 2 by presenting the theoretical background. It is based on the notion due to Richard Parker of local submodules and closely related algebra elements, called *peakwords*. We restate their most important properties by citing results of [6], and present how they may be used to solve the matching problem mentioned above. In Section 3 we formulate and discuss an algorithm which implements the Matching Lemma of Section 2. Section 4 illustrates how our method may be applied in the case that the algebra  $A$  is a group algebra of some finite group  $G$ . We exemplify its usability for permutation modules and homomorphism spaces, respectively tensor products of modules. Finally, in

Section 5 we witness the proposed algorithm in action by presenting a practical example, namely a tensor product of simple modules for the sporadic Harada–Norton group HN.

## 2. $S$ -local submodules and matching

Closely related to the composition factors of an  $A$ -module  $V$  are its so-called *local* submodules. As our approach to resolve the matching problem relies on the properties of these submodules, we recall some basic facts about local submodules and their relationship with *peakwords* as introduced in [6].

Let  $S$  be a simple  $A$ -module. An  $A$ -module is called  $S$ -local, if it possesses a unique maximal submodule such that the corresponding factor module is isomorphic to  $S$ . By [6, Theorem 2.3], the  $S$ -local submodules of  $V$  have the nice property that generators for each may be obtained by condensing  $V$  with an  $S$ -primitive idempotent, i.e., with a primitive idempotent corresponding to the simple module  $S$ . In general it is a difficult problem to determine such an idempotent. But, as shown in [6], it may be resolved by considering certain elements of the algebra  $A$  called *peakwords*.

For an element  $a \in A$ , we denote the kernel, respectively the image, of the action of  $a$  on an  $A$ -module  $M$  by  $\ker_M(a)$ , respectively  $\text{im}_M(a)$ . Then, if  $V$  is a faithful  $A$ -module and  $S$  a composition factor of  $V$ , an element  $a \in A$  is called an  $S$ -peakword, if  $\ker_T(a) = 0$  for all composition factors  $T$  of  $V$  which are not isomorphic to  $S$ , and  $\dim_F(\ker_S(a^2))$  is equal to the degree of the splitting field of  $S$ .

Given an  $S$ -peakword  $w \in A$  for a faithful module  $V$ , the fitting decomposition of  $V$  with respect to  $w$ , i.e., the decomposition  $V = \ker_V(w^N) \oplus \text{im}_V(w^N)$  for some sufficiently large  $N \in \mathbb{N}$ , gives rise to the condensation of  $V$  with a uniquely determined  $S$ -primitive idempotent  $f$  via  $Vf = \ker_V(w^N)$  (see [6, Theorems 3.1 and 3.4] for details).

With the  $S$ -local submodules thus available to us, we proceed to show how they provide the necessary means to solve the matching problem.

Keeping the paradigm of condensation in mind which we mentioned in the introduction, we will not seek peakwords within the original algebra, but instead within a condensed algebra. The key idea now is that we may lift a local submodule of a condensed module to a local submodule of the original module as stated in Lemma 2.1.

For brevity, we will subsequently write  $\ker_M(w^\infty)$  to denote the stable kernel of some power of a word  $w$  on a module  $M$ .

**LEMMA 2.1.** *Let  $V$  be a faithful  $A$ -module,  $S$  a simple  $A$ -module, and  $e \in A$  an idempotent with  $Se \neq 0$ . Furthermore let  $w \in eAe$  be an  $Se$ -peakword and  $0 \neq v \in \ker_{Ve}(w^\infty)$ . Then  $vA$  is an  $S$ -local submodule of  $V$ .*

*Proof.* We have that  $\ker_{Ve}(w^\infty) = Vef$  for some  $Se$ -primitive idempotent  $f \in eAe$ . Therefore  $ef = fe = f$ , and  $feAef = fAf$  is local. Hence  $f$  is primitive as an idempotent of  $A$ , and since  $0 \neq Sef = Sf$ , it is  $S$ -primitive. Thus  $vA$  is an  $S$ -local submodule of  $V$ . □

The local submodules also provide us with a natural criterion to determine if a chosen condensation idempotent annihilates a particular simple module.

LEMMA 2.2. *Let  $v \in V$  such that  $vA$  is  $S$ -local, and  $e \in A$  an idempotent. Then  $veA = vA$  if and only if  $Se \neq 0$ .*

*Proof.* As the  $A$ -module generated by  $v + \text{rad}(vA)$  is isomorphic to  $S$ , we have that  $ve \in \text{rad}(vA)$  if and only if  $Se = 0$ . Hence  $veA$  is a proper submodule of  $vA$  if and only if  $Se = 0$ , and conversely  $veA = vA$  if and only if  $Se \neq 0$ .  $\square$

Combining the two previous lemmas, we may now formulate our solution to the matching problem.

LEMMA 2.3 (Matching Lemma). *Let  $V$  be a faithful  $A$ -module,  $S$  a simple  $A$ -module, and  $e, e' \in A$  idempotents with  $Se \neq 0$ . Furthermore let  $w \in eAe$  be an  $Se$ -peakword, and  $0 \neq v \in \ker_{Ve}(w^\infty)$ .*

*We have  $ve'eAe = veAe$ , if and only if  $Se' \neq 0$ .*

*If  $Se' \neq 0$ , then  $ve'Ae'$  is an  $Se'$ -local module.*

*Proof.* Let  $ve'eAe = veAe$ . Then  $ve' \notin \text{rad}(vA)$ , as otherwise  $ve'A$  is a proper submodule of  $vA$ , which implies  $(vA)e \geq (ve'A)e \geq ve'eAe = veAe = (vA)e$  under the hypothesis, and by Lemmas 2.1 and 2.2. Therefore  $ve'A = vA$ , and Lemma 2.2 gives  $Se' \neq 0$ . Finally, since  $ve'Ae' = vAe'$ , and as  $Se' \neq 0$ , the module  $ve'Ae'$  is  $Se'$ -local, of course.

For the converse observe that by Lemma 2.2 both  $ve'$  and  $ve$  fulfill  $ve'A = vA = veA$ . Since therefore  $ve'$  generates an  $S$ -local module, applying the lemma again, we obtain  $ve'eA = ve'A = vA$ , which is  $veA$ .  $\square$

From Lemma 2.3 it is now clear how we may match two simple modules for the condensed algebras  $eAe$  and  $e'Ae'$ : If the condition of Lemma 2.3 is met, we obtain a link between the simple  $eAe$ -module  $Se$  and the simple  $e'Ae'$ -module  $Se'$  by determining the head of the  $Se'$ -local  $e'Ae'$ -module generated by  $ve'$ . On the other hand, if the condition is not fulfilled, we may conclude that  $Se$  cannot be matched to any simple  $e'Ae'$ -module.

### 3. From theory to practice: An algorithm

Let  $V$  be a faithful  $A$ -module, and  $e, e'$  idempotents in  $A$ .

In practice we are given  $Ve$  and  $Ve'$  in terms of matrices representing the action of generators of  $eAe$  and  $e'Ae'$  on  $Ve$  and  $Ve'$ , respectively. Therefore a run of the Meat-Axe will provide us with the composition factors of  $Ve$ , respectively  $Ve'$ . Let  $S$  be a composition factor of  $Ve$ . Then in order to match  $S$  to a composition factor  $S'$  of  $Ve'$ , it is clear from Lemma 2.3 that we may apply Algorithm 1 whose practical implementation we scrutinize in the following.

The first step in the algorithm presented is to determine a peakword for the simple module  $S$ . Its practical realization is straightforward (see also [6, Section 5]): We conduct a peakword search in the algebra  $eAe$  by pseudo-randomly generating words in  $eAe$ , and checking, if one of them fulfills the defining properties of a peakword. It then needs to be evaluated on the whole condensed module  $Ve$ .

---

**Algorithm 1** Matching a composition factor.

---

**Input:** A composition factor  $S$  of  $Ve$  and a database DB of isotypes of composition factors of  $Ve'$ .

**Output:** The  $S'$  in DB which matches  $S$ , or **fail** if no match is possible.

- 1: Determine an  $S$ -peakword  $w \in eAe$  and compute the the kernel of its action on  $Ve$ , i.e.,  $\ker_{Ve}(w)$ .
  - 2: Embed any  $0 \neq v \in \ker_{Ve}(w)$  into  $V$ .
  - 3: Project  $v$  onto  $Ve'$ , obtaining  $ve' \in Ve'$ .
  - 4: Embed  $ve'$  into  $V$ .
  - 5: Project  $ve'$  onto  $Ve$ , giving  $ve'e \in Ve$ .
  - 6: **if**  $\dim_F(ve'eAe) = \dim_F(veAe)$  **then**
  - 7:     Compute the head  $H$  of  $ve'Ae'$ .
  - 8:     Find the module  $S'$  in DB isomorphic to  $H$ .
  - 9:     **return**  $S'$ .
  - 10: **else**
  - 11:     **return** **fail**.
  - 12: **end if**
- 

Note that, since by Lemma 2.3 any arbitrary nonzero vector in the stable kernel of a peakword is sufficient for our matching purposes, it is not necessary to compute the full stable kernel.

As the computation of the kernel  $\ker_{Ve}(w)$  is conducted within the condensed module  $Ve$ , the nonzero kernel vector  $v$  chosen in Line 2 is given with respect to a basis of the condensed space  $Ve$ . Therefore to embed  $v$  into  $V$ , we have to *uncondense* it. This is done by choosing a preimage  $\hat{v} \in V$  of  $v$  under the projection epimorphism onto  $Ve$  induced by the right-multiplication with  $e$ .

In Line 3 of Algorithm 1 we project the uncondensed vector  $\hat{v} \in V$  through right-multiplication with the idempotent  $e'$ . This can be achieved while eschewing an explicit application of  $e'$  in the usually huge space  $V$ , by applying the projection part of the algorithm used to condense  $V$  to  $Ve'$ . As indicated in the introduction, the efficacy of condensation algorithms usually depends on a special basis of  $V$ . Therefore in preparation of this projection step, we need to change the basis with respect to which  $\hat{v}$  is given to the special basis of  $V$  underlying the condensation algorithm. Hence, it is inevitable that we do one, albeit comparatively small, calculation explicitly in  $V$ . For practical applications it is therefore crucial that this calculation may be realized. We illustrate in the following section that this obstacle can be overcome for a large class of problems encountered in practice.

In Lines 4 and 5 we complete our zig-zag approach by repeating the procedures of Lines 2 and 3 with the roles of the idempotents  $e$  and  $e'$  interchanged.

The remaining Lines 6 to 12 rely only on standard Meat-Axe functionality. To verify the condition that the dimensions of both spaces  $ve'eAe$  and  $veAe$  are equal, it suffices to construct bases of both  $eAe$ -modules using the Meat-Axe's spinning algorithm. As we work with modules of the condensed algebra  $eAe$ , we can expect their dimensions to be computationally manageable. If we are able to verify equality of dimensions, we proceed to compute the head of  $ve'Ae'$  (see [9]), and compare it with the database of  $e'Ae'$ -composition factors of  $Ve'$ . On the other hand, if the

condition of Line 6 cannot be verified, Lemma 2.3 gives that the simple  $A$ -module  $\hat{S}$  for which  $\hat{S}e \cong S$  is annihilated by  $e'$  and thus there is no possible match, and a return value of `fail` is given.

#### 4. Matching in practice: Condensed group algebras

As related in the introduction, we are primarily interested in the applications of condensation in the computational representation theory of finite groups, and in the Modular Atlas project in particular. Therefore in this section we take  $A$  to be the group algebra  $FG$  for some finite group  $G$ .

To demonstrate the usability of the Matching Lemma in practice, we detail how the unavoidable calculations in the uncondensed original module  $V$ , i.e., the projections of Lines 3 and 5 of Algorithm 1, and in particular the necessary basis changes, may be contrived for two important condensation methods: The condensation of tensor products, or homomorphism spaces, and the condensation of permutation modules.

In the case of tensor products it is important to note that given two modules  $V$  and  $W$  of dimensions  $n$  and  $m$ , and affording (matrix) representations  $\rho_V$  and  $\rho_W$ , then even when computing in the tensor product  $V \otimes W$ , we never need to work with  $nm \times nm$ -representing matrices. Instead, we make use of the natural isomorphism  $V \otimes W \cong \text{Hom}_F(V^*, W)$  (see [8, Lemma 2.6]): The action of a group element  $g \in G$  on a vector  $x \in V \otimes W$  may be realized by *furling*  $x$  into an  $n \times m$ -matrix  $X$ , i.e., applying the isomorphism  $\Phi$  of [8, Lemma 2.6], evaluating  $\rho_V(g)^{\text{tr}} \cdot X \cdot \rho_W(g)$ , and *unfurling* the matrix  $X$  to an element of  $V \otimes W$  again, i.e., applying  $\Phi^{-1}$ . Therefore, from a computational point of view, it is more natural to consider homomorphism spaces of modules, whose condensation can be handled very efficiently (see [7]).

Furthermore, as the basis of the tensor product is derived from the bases of the factors  $V$  and  $W$ , a change of basis with respect to which a vector  $x \in V \otimes W$  is given, is also easily realized by multiplying the furled up  $x$  from the left and from the right with appropriate base-change matrices of the factors. The latter are computed prior to condensation (see [8] for details) giving us matrices  $B \in F^{n \times n}$  and  $C \in F^{m \times m}$ , such that any  $x \in V \otimes W$  may be rewritten with respect to the new basis by evaluating  $\Phi^{-1}(B^{-\text{tr}} \cdot \Phi(v) \cdot C^{-1})$ .

Hence, since all computations within  $V \otimes W$  are conducted with furled-up elements, the uncondensed vector of a peakword kernel is given as an  $n \times m$ -matrix, and it only takes two matrix multiplications in the sizes of the tensor factors, before the projection part of the condensation algorithm can be applied.

Let us consider permutation modules next. Here, the necessary basis change may be obtained by ‘merely’ permuting the entries of an uncondensed vector. Of course, since in current applications the considered permutation modules possess dimensions which range in the billions, this is a nontrivial task. The solution to this problem is to never explicitly uncondense a peakword kernel vector in the first place.

The requisites for condensing a permutation module  $V$  with basis  $\Omega$  are the orbits of the condensation subgroup on  $\Omega$  (see, for example, [5, Section 5]). For simplicity, we take the condensation idempotents to be of the form  $e = \frac{1}{|K|} \sum_{k \in K} k$ , if  $K$  is a condensation subgroup, i.e., the canonical basis of  $Ve$  is given by the  $K$ -orbit sums

on  $\Omega$ . Then given two condensation subgroups  $K$  and  $K'$ , and an element  $v \in Ve$ , the task of determining the coefficients of  $ve'$  with respect to the canonical basis of  $Ve'$ , relies on the information in which  $K'$ -orbit a point of a  $K$ -orbit lies.

This information may be gained at little extra cost: Condensing with the idempotent  $e$ , for example, the algorithm loops over every point of all  $K$ -orbits on  $\Omega$ , and therefore in every step we may simultaneously look up in which  $K'$ -orbit the point lies. Hence the data structure necessary for the projection with  $e'$  is constructed just once during the condensation of any one element at the cost of one such look-up for every point of  $\Omega$ .

Then, as given in Lines 2 and 3 of Algorithm 1, the uncondensation of  $v$  and its projection to  $ve'$  may be conducted as in Algorithm 2, for which we let  $O_1, \dots, O_n$  and  $O'_1, \dots, O'_{n'}$  denote the  $K$ - and  $K'$ -orbit sums on  $\Omega$ , respectively.

---

**Algorithm 2** Uncondensation and projection procedure for permutation modules

---

**Input:** The coefficient vector  $[v_1, \dots, v_n]$  of an element  $v = \sum_{i=1}^n v_i O_i \in Ve$ , and the data `WhichOrbit`, which for an  $\omega \in \Omega$  gives the number of its  $K'$ -orbit.

**Output:** The coefficient vector  $[z_1, \dots, z_{n'}]$  of  $ve' = \sum_{i=1}^{n'} z_i O'_i \in Ve'$ .

- 1: Set  $z := 0 \in F^{n'}$ .
  - 2: **for**  $1 \leq i \leq n$  **do**
  - 3:   **for** every  $\omega \in O_i$  **do**
  - 4:      $j := \text{WhichOrbit}(\omega)$ .
  - 5:      $z_j := z_j + \frac{v_i}{|O_j|}$
  - 6:   **end for**
  - 7: **end for**
  - 8: **return**  $z$ .
- 

While we still need to process all points in the  $G$ -set  $\Omega$ , this way only one  $K$ -orbit needs to be stored in memory at a time. Also, the calculation is shortened by only considering the  $K$ -orbits which correspond to nonzero entries of the coefficient vector of  $v$ . Furthermore, as every  $K$ -orbit may be treated independently of all other  $K$ -orbits, this procedure is trivially parallelizable, and is easily distributed to several machines. Thus Algorithm 1 is viable even for large permutation modules.

### 5. An example: HN in characteristic 3

The purpose of this section is to substantiate our previous theoretical discussion with a practical example. To this end we consider the tensor product  $V := 133a \otimes 8778a$  of irreducible representations of the sporadic Harada–Norton group  $G := \text{HN}$  over a field  $F$  of characteristic 3. Our motivation for considering this tensor product will be detailed in a forthcoming paper in which the 3-modular characters of HN are classified as a contribution to the Modular Atlas project.

The information of an  $FG$ -module's composition factors and their multiplicities is naturally essential to gain insight into the irreducible modular characters of  $G$ , and several variations of this theme have been applied successfully in the past. See, for example, [4] and [12] how the simple modules of a group algebra may be inferred by studying several modules through condensation techniques.

As condensation subgroups for  $133a \otimes 8778a$  we choose the normal 2- and 5-subgroups of order 2048 and 3125 contained in the ninth and tenth maximal subgroups (in Atlas notation)  $2^3.2^2.2^6.(3 \times L_3(2))$  and  $5^2.5.5^2.4A_5$  of  $G$ , respectively. We denote the trace idempotents to which both groups give rise by  $e$  and  $e'$ , respectively, and accordingly distinguish  $eFGe$ -modules from  $e'FGe'$ -modules by affixing prime marks to the latter.

Condensing  $V$  with both idempotents yields the module  $Ve$  of dimension 567 and the module  $Ve'$  whose dimension is 438. Note that, while neither  $e$  nor  $e'$  is faithful, no simple  $FG$ -module condenses to zero under both idempotents. Applying the Meat-Axe, we obtain the following decompositions into composition factors, in which we denote a condensed module with a lower case  $k$  for emphasis:

$$\begin{aligned}
 Ve &= 5 \times k1a + k1b + 2 \times k8a + 2 \times k9a + 2 \times k26a + k26b & (1) \\
 &\quad + 2 \times k138a + k173a, \\
 Ve' &= k1a' + 2 \times k3a' + 2 \times k7a' + 3 \times k8a' + 2 \times k8b' + & (2) \\
 &\quad + 3 \times k8c' + k8d' + 4 \times k16a' + k89a' + 2 \times k96a'.
 \end{aligned}$$

We may now match the composition factors of either decomposition (1) or (2) to the constituents of the other decomposition: we either consider the composition factors of  $Ve$ , and find their matches in the composition factors of  $Ve'$ , i.e., we match *from  $Ve$  to  $Ve'$* , or vice versa. As the majority of calculations in Algorithm 1 occurs within the module whose composition factors we are trying to match, in practice it is sensible to match from the smaller dimensional module to the larger dimensional one. For illustrative purposes, however, we apply our GAP implementation of Algorithm 1 to match both ways in this example. The output matchings produced are given in Tables 1 and 2.

On a machine with an AMD Athlon processor running at 2400Mhz the matching of a simple module takes around 40 seconds either way. Most of this time (approx. 30 seconds) is spent on computing the basis changes in the uncondensed modules, while the remaining time is mainly accounted for by spinning and determining the action on submodules. However, we expect that, as the dimension of the condensed modules increases, the time spent on the calculations in the condensed module will eventually exceed the time needed for the basis changes in the uncondensed module.

Table 1: Matching the composition factors of  $(133a \otimes 8778a)e$  to composition factors of  $(133a \otimes 8778a)e'$

$Ve$	$k1a$	$k1b$	$k8a$	$k9a$	$k26a$	$k26b$	$k138a$	$k173a$
$Ve'$	—	$k1a'$	$k7a'$	—	$k8b'$	$k8d'$	$k96a'$	$k89a'$

Table 2: Matching the composition factors of  $(133a \otimes 8778a)e'$  to composition factors of  $(133a \otimes 8778a)e$

$Ve'$	$k1a'$	$k3a'$	$k7a'$	$k8a'$	$k8b'$	$k8c'$	$k8d'$	$k16a'$	$k89a'$	$k96a'$
$Ve$	$k1b$	—	$k8a$	—	$k26a$	—	$k26b$	—	$k173a$	$k138a$

Finally, considering Decompositions (1) and (2) together with the matching of Table 1, we can now give the entire decomposition of  $V$  into composition factors as

$$V = 5 \times \widehat{k1a} + \widehat{k1a'} + 2 \times \widehat{k3a'} + 2 \times \widehat{k7a'} + 3 \times \widehat{k8a'} + 2 \times \widehat{k8b'} + \\ + 3 \times \widehat{k8c'} + \widehat{k8d'} + 2 \times \widehat{k9a} + 4 \times \widehat{k16a'} + \widehat{k89a'} + 2 \times \widehat{k96a'},$$

in which the hats above the simple modules indicate that the sum is taken over their respective origins in  $\text{mod-}FG$ , as defined in the introduction.

### References

1. ‘The Modular Atlas Homepage’, [www.math.rwth-aachen.de/~MOC/](http://www.math.rwth-aachen.de/~MOC/). 213
2. J. H. CONWAY, R. T. CURTIS, S. P. NORTON, R. A. PARKER and R. A. WILSON, *Atlas of finite groups*. Maximal subgroups and ordinary characters for simple groups. With computational assistance from J. G. Thackray (Oxford University Press, Eynsham, 1985). ISBN 0-19-853199-0. 213
3. J. A. GREEN, *Polynomial representations of  $GL_n$* , Lecture Notes in Mathematics 830 (Springer, Berlin, 1980). ISBN 3-540-10258-2. 213
4. G. HISS, M. NEUNHÖFFER and F. NOESKE, ‘The 2-modular characters of the Fischer group  $Fi_{23}$ ’, *J. Algebra* 300 (2006) 555–570. 219
5. F. LÜBECK and M. NEUNHÖFFER, ‘Enumerating large orbits and direct condensation’, *Experiment. Math.* 10 (2001) 197–205. 214, 218
6. K. LUX, J. MÜLLER and M. RINGE, ‘Peakword condensation and submodule lattices: an application of the MEAT-AXE’, *J. Symbolic Comput.* 17 (1994) 529–544. 214, 215, 216
7. K. LUX, M. NEUNHÖFFER and F. NOESKE, ‘Condensation of homomorphism spaces’, in preparation. 214, 218
8. K. LUX and M. WIEGELMANN, ‘Condensing tensor product modules’, *The atlas of finite groups: ten years on*, Birmingham, 1995, London Math. Soc. Lecture Note Ser. 249 (Cambridge Univ. Press, Cambridge, 1998) 174–190. 214, 218
9. K. LUX and M. WIEGELMANN, ‘Determination of socle series using the condensation method’, *Computational algebra and number theory*, Milwaukee, WI, 1996, *J. Symbolic Comput.* 31 (2001) 163–178. 217
10. J. MÜLLER and J. ROSENBOOM, ‘Condensation of induced representations and an application: the 2-modular decomposition numbers of  $Co_2$ ’, *Computational methods for representations of groups and algebras*, Essen, 1997, Progr. Math. 173 (Birkhäuser, Basel, 1999) 309–321. 214
11. F. NOESKE, ‘Morita-Äquivalenzen in der algorithmischen Darstellungstheorie’, PhD thesis, RWTH Aachen University, 2005. 214
12. F. NOESKE, ‘The 2- and 3-modular characters of the sporadic simple Fischer group  $Fi_{22}$  and its cover’, *J. Algebra* 309 (2007) 723–743. 219
13. R. A. PARKER, ‘The computer calculation of modular characters (the Meat-Axe)’, *Computational group theory*, Durham, 1982 (Academic Press, London, 1984) 267–274. 213

14. A. J. E. RYBA, ‘Computer condensation of modular representations. Computational group theory, Part 1, *J. Symbolic Comput.* 9 (1990) 591–600. [213](#)
15. A. J. E. RYBA, ‘Condensation of symmetrized tensor powers’, *J. Symbolic Comput.* 32 (2001) 273–289. [214](#)

Felix Noeske [Felix.Noeske@math.rwth-aachen.de](mailto:Felix.Noeske@math.rwth-aachen.de)

Lehrstuhl D für Mathematik  
RWTH Aachen University  
52056 Aachen, Germany