Normal Forms for Operators via Gröbner Bases in Tensor Algebras

Jamal Hossein Poor, Clemens G. Raab, and Georg Regensburger(✉)

Johann Radon Institute for Computational and Applied Mathematics (RICAM),
Austrian Academy of Sciences, Linz, Austria
`{jamal.hossein.poor,clemens.raab,georg.regensburger}@ricam.oeaw.ac.at`

Abstract. We propose a general algorithmic approach to noncommutative operator algebras generated by linear operators using quotients of tensor algebras. In order to work with reduction systems in tensor algebras, Bergman's setting provides a tensor analog of Gröbner bases. We discuss a modification of Bergman's setting that allows for smaller reduction systems and tends to make computations more efficient. Verification of the confluence criterion based on S-polynomials has been implemented as a Mathematica package. Our implementation can also be used for computer-assisted construction of Gröbner bases starting from basic identities of operators. We illustrate our approach and the software using differential and integro-differential operators as examples.

Keywords: Operator algebra · Tensor algebra · Noncommutative Gröbner basis · Reduction systems

1 Introduction

For an algorithmic treatment of many common operator algebras, like differential and difference operators, skew polynomials are a well-established algebraic construction; see e.g. the survey [4] describing also implementations in computer algebra systems. However, not all common operator algebras are covered by this setting. For example, integral operators cannot be constructed that way.

The principle that can always be applied is construction by generators and relations. In practice, normal forms are needed for effective computation. Finding and proving the structure of normal forms is a difficult task, the general problem is even undecidable. For skew polynomials, normal forms are given by the standard polynomial basis. For tensor algebras, Bergman's paper [1] also provides a framework in which reduction systems and corresponding normal forms can be analyzed, analogous to Gröbner bases. Tensor algebras can be seen as a generalization of free noncommutative polynomial algebras and inherit all their algorithmic obstructions. At the same time, parts of the tensor setting can be automated, in particular, verification of the confluence criterion and subsequent

---

All authors were supported by the Austrian Science Fund (FWF): P27229.

G.-M. Greuel et al. (Eds.): ICMS 2016, LNCS 9725, pp. 505–513, 2016.
DOI: 10.1007/978-3-319-42432-3_65

computations with normal forms. We provide an implementation in the Mathematica package `TenReS` including our generalization of Bergman's setting [6]: http://gregensburger.com/software/TenReS.zip.

When representing linear operators as tensors, composition of operators is modeled by the tensor product. We briefly describe the main building blocks of the algebraic construction. Over some K-module of basic operators M, we consider the tensor algebra

$$K\langle M\rangle = \bigoplus_{n=0}^{\infty} M^{\otimes n},$$

where K is commutative ring with unit element. Operator identities are encoded as a reduction system Σ for $K\langle M\rangle$. Then the operator algebra is constructed as

$$K\langle M\rangle / I_{\Sigma},$$

where I_{Σ} is a two-sided ideal induced by the reduction system Σ. This setting allows for finite reduction systems even in cases where the module M does not have a finite basis. As our examples illustrate, this approach does not make use of a basis of M at all.

2 Reduction Systems for Tensor Algebras

Using the well-known example of differential operators, we explain the main theoretical notions as well as the most important commands of our package. Usually one defines the differential operators directly via skew polynomials with normal forms $\sum f_i\, \partial^i$ and noncommutative multiplication defined by

$$\partial f = f\partial + f'.$$

Suppose we do not already know the normal forms of differential operators and we just have the definition of the derivation ∂ as a K-linear operator on some differential K-algebra (R, ∂) obeying the Leibniz rule $\partial fg = (\partial f)g + f\partial g$. (Note that we use operator notation in this paper.) Recall that differential operators with polynomial coefficients (Weyl algebra) over a field $K \supseteq \mathbb{Q}$ can be defined as the quotient algebra $K\langle X, D\rangle/(DX - XD - 1)$ of the free noncommutative polynomial algebra $K\langle X, D\rangle$ modulo the two-sided ideal $(DX - XD - 1)$. Now, we want to do an analogous construction for the differential operators with coefficients in an arbitrary differential K-algebra (R, ∂). To this end, we work in the tensor algebra over the K-module $M = R \oplus K\partial$. First, we explain some main points informally before explaining necessary technical details later.

We interpret elements $f \in R$ as multiplication operators, ∂ as the derivative operator on R. Then we have three basic identities between these operators: Multiplication by $1 \in R$ acts like applying no operator at all. So we can replace it by the empty tensor ϵ, which represents the unit element in the tensor algebra. For $f, g, h \in R$ with $fg = h$ the multiplication operators

$$f \otimes g \quad \text{and} \quad h$$

act in the same way on R. Finally, the Leibniz rule implies that the operators

$$\partial \otimes f \quad \text{and} \quad f \otimes \partial + \partial f$$

act in the same way as well. Deciding to move the differential operators to the right results in the following reduction rules to simplify parts of tensors representing a composition of multiplication operators and the derivative operator:

$$1 \mapsto \epsilon, \quad f \otimes g \mapsto fg, \quad \text{and} \quad \partial \otimes f \mapsto f \otimes \partial + \partial f.$$

A priori it is not clear that we will always end up with the same result if we apply the reduction rules in different ways. Suppose we have a tensor of the form

$$f_1 \otimes f_2 \otimes f_3$$

with $f_1, f_2, f_3 \in R$ arbitrary but fixed. Then we can apply the reduction rule for composition of multiplication operators in different ways obtaining $(f_1 f_2) \otimes f_3$ and $f_1 \otimes (f_2 f_3)$. Trivially, another application of the same rule yields $f_1 f_2 f_3$ in both cases. In general, a minimal case where two (not necessarily distinct) rules can be applied differently to a tensor is called an *ambiguity* and the difference

$$(f_1 f_2) \otimes f_3 - f_1 \otimes (f_2 f_3)$$

is called the corresponding *S-polynomial*. If all S-polynomials of an ambiguity can be reduced to zero by the reduction rules, like above for all f_1, f_2, f_3, then we call the ambiguity *resolvable*.

Analogous to Buchberger's criterion for Gröbner bases [3] and the Composition-Diamond Lemma for Gröbner-Shirshov bases [2], we have a confluence criterion for tensor reduction systems due to Bergman [1]. A reduction system defines unique normal forms if and only if all ambiguities are resolvable. Termination of the reduction process, depends on a compatible Noetherian ordering on words; see [1,6] for details. Throughout this paper and in the package, we tacitly assume that such an ordering exists.

One can distinguish four different types of ambiguities: overlaps and inclusions, each with or without specialization. The ambiguity above is an overlap ambiguity since the factors $f_1 \otimes f_2$ and $f_2 \otimes f_3$ of the tensor $f_1 \otimes f_2 \otimes f_3$ on which the rules act overlap. There is no specialization involved since all cases on which the rules may act actually arise in this way. On the other hand, the tensor

$$\partial \otimes c$$

with $c \in R$ and $\partial c = 0$ may be reduced by the rules $1 \mapsto \epsilon$ or $\partial \otimes f \mapsto f \otimes \partial + \partial f$ to either $c\partial$ or $c \otimes \partial$. Obviously, another application of $1 \mapsto \epsilon$ in the second case gives $c\partial$ as well, so this ambiguity is resolvable again. In this case one factor c of $\partial \otimes c$ on which one rule acts is contained in the other factor $\partial \otimes c$ acted on by the other rule. So we call it an inclusion ambiguity. Moreover, not all cases of the rule $\partial \otimes f \mapsto f \otimes \partial + \partial f$ are needed here, just $f \in K \subsetneq R$, so we say that this ambiguity involves specialization.

2.1 Tensor Algebras

For computation in the tensor algebra $K\langle M\rangle$ over the K-module M, we need to be able to check when objects are contained in K. The user has to implement the function `CoeffQ` returning `True` whenever its argument is contained in K. Based on that, tensors $m_1\otimes\cdots\otimes m_n \in K\langle M\rangle$ are represented as $\mathtt{Prod}[\mathtt{m_1},\ldots,\mathtt{m_n}]$ respecting K-multilinearity of the tensor product.

In order to fix domains for the reduction rules, we need corresponding direct sum decompositions of the module M indexed by some finite set (alphabet). In our example above, we define $M_F = R$ and $M_D = K\partial$ so that

$$M = M_F \oplus M_D.$$

Hence, in terms of the word monoid $\langle Y\rangle$ over the alphabet $Y = \{F, D\}$, we have the following decomposition of the tensor algebra into modules $M_W = M_{y_1}\otimes\cdots\otimes M_{y_n}$ for $W = y_1 \ldots y_n \in \langle Y\rangle$:

$$K\langle M\rangle = \bigoplus_{W\in\langle Y\rangle} M_W.$$

For each submodule M_F and M_D of M, the user has to implement a membership test $\mathtt{MemberQ_F}$ and $\mathtt{MemberQ_D}$ indexed by the corresponding letter of the alphabet. Moreover, the user has to implement all computations with elements of each module. In particular, this applies to additional operations on a module, like in our example multiplication and derivation on M_F respecting the Leibniz rule in R. For computation with S-polynomials, we need to compute with general elements of non-cyclic modules. For example, in the module M_F, general elements will always be called `F[1]`, `F[2]`, …, which together with their derivatives also have to be recognized by the membership test.

In order to formalize the rule $1 \mapsto \epsilon$, we need a refinement of the above decomoposition of M by choosing a complement $M_{\tilde{F}}$ of $M_K = K$ such that

$$M_F = M_K \oplus M_{\tilde{F}}.$$

Of course, also for the new submodules, membership tests have to be implemented. All such refinements of submodules have to be stored in the variable

$$\mathtt{Specialization=}\{\mathtt{F}\rightarrow\{\mathtt{K,\tilde{F}}\}\}.$$

Altogether, with $X = \{K, \tilde{F}, D\}$ we have two decompositions of M:

$$M = \bigoplus_{x\in X} M_x = \bigoplus_{y\in Y} M_y.$$

For all cyclic submodules the user should store the letter and the generator as a pair in the list `CyclicModules`. In our example we denote ∂ by `Diff`, so that

$$\mathtt{CyclicModules=}\{\{\mathtt{K,1}\},\{\mathtt{D,Diff}\}\}.$$

2.2 Reduction Systems

With the submodules introduced above, the reduction rule $f \otimes g \mapsto fg$ can be formally defined as the pair (FF, h_{FF}) consisting of the word $FF \in \langle Y \rangle$ and the module homomorphism $h_{FF}\colon M_{FF} \to M_F \subset K\langle M \rangle$. The homomorphism h_{FF} is defined by $f \otimes g \mapsto fg$ and could be implemented by the user as $\mathtt{h}_{\mathtt{FF}}[\mathtt{f\_}, \mathtt{g\_}] := \mathtt{Prod}[\mathtt{mul}[\mathtt{f}, \mathtt{g}]]$, for example. Similarly, we have the reduction rule (DF, h_{DF}), where $h_{DF}\colon M_{DF} \to M_{FD} \oplus M_F$ may be implemented as $\mathtt{h}_{\mathtt{DF}}[\mathtt{Diff}, \mathtt{f\_}] := \mathtt{Prod}[\mathtt{f}, \mathtt{Diff}] + \mathtt{Prod}[\mathtt{Diff}[\mathtt{f}]]$. Thanks to the refinement, we also can define the reduction rule $1 \mapsto \epsilon$ as the pair (K, h_K) with the homomorphism $h_K\colon M_K \to M_\epsilon$ ($\mathtt{h}_K[1] := \mathit{Prod}[]$), where ϵ is the empty word. These homomorphisms have to be implemented by the user.

Our reduction system over the *combined alphabet* $Z = X \cup Y$ is given by

$$\Sigma = \big\{(FF, f \otimes g \mapsto fg), \quad (DF, \partial \otimes f \mapsto f \otimes \partial + \partial f), \quad (K, 1 \mapsto \epsilon)\big\}$$

The corresponding definition for our package is

$$\mathtt{RedSys} = \{\{\{\mathtt{F}, \mathtt{F}\}, \mathtt{h}_{\mathtt{FF}}\}, \{\{\mathtt{D}, \mathtt{F}\}, \mathtt{h}_{\mathtt{DF}}\}, \{\{\mathtt{K}\}, \mathtt{h}_{\mathtt{K}}\}\}.$$

In general, any reduction rule $r = (W, h)$, where h is a K-module homomorphism $h\colon M_W \to K\langle M \rangle$ reduces tensors $a \otimes w \otimes b$ with $a \in M_A$, $w \in M_W$, and $b \in M_B$ for some $A, B \in \langle Z \rangle$ by $a \otimes w \otimes b \to_r a \otimes h(w) \otimes b$. In other words, similarly to polynomial reduction, we "replace" the "monomial" w by the "tail" $h(w)$ given by the homomorphism h. The *reduction ideal* induced by a reduction system Σ is defined as the two-sided ideal

$$I_\Sigma := (t - h(t) \mid (W, h) \in \Sigma \text{ and } t \in M_W) \subseteq K\langle M \rangle.$$

If one tensor can be reduced to another, then their difference is contained in I_Σ. This is implemented via the command `ApplyRules`. For example, we can reduce $f_1 \otimes \partial \otimes f_2$ to $(f_1 f_2) \otimes \partial + f_1 \partial f_2$ by the reflexive-transitive closure $\overset{*}{\to}_\Sigma$:

```
ApplyRules[Prod[F[1], Diff, F[2]], RedSys]
Prod[mul[F[1], Diff[F[2]]]] + Prod[mul[F[1], F[2]], Diff]
```

2.3 Normal Forms and Confluence

Determination of normal forms and ambiguities can be reduced to problems in the word monoid $\langle Z \rangle$ over the combined alphabet. We introduce the notion of specializing a word $W \in \langle Z \rangle$ to a word in $\langle X \rangle$ by replacing all letters of W from $Y \setminus X$ by corresponding letters from X. For example, the specializations of the word $FFD \in \langle Z \rangle$ are given by $\{KKD, K\tilde{F}D, \tilde{F}KD, \tilde{F}\tilde{F}D\} \subset \langle X \rangle$. In terms of modules, we then have that M_W is the direct sum of all M_V such that V is a specialization of W.

The module $K\langle M \rangle_{\text{irr}}$ of *irreducible tensors*, i.e. tensors that cannot be reduced by any reduction rule in Σ, is determined by the set of *irreducible words* $\langle X \rangle_{\text{irr}} \subseteq \langle X \rangle$ via

$$K\langle M \rangle_{\text{irr}} = \bigoplus_{W \in \langle X \rangle_{\text{irr}}} M_W.$$

Irreducible words are those words over the refined alphabet X that do not contain a subword that is a specialization of the word W of any rule $(W, h) \in \Sigma$. In our example one easily sees that the irreducible words are given by D^j and $\tilde{F}D^j$ with $j \in \mathbb{N}_0$. Consequently, irreducible tensors are of the form $\partial^{\otimes j}$ and $f \otimes \partial^{\otimes j}$ with $j \in \mathbb{N}_0$ and $f \in M_{\tilde{F}}$ and K-linear combinations thereof.

In order to show that irreducible tensors already define a normal form of tensors in $K\langle M \rangle$ modulo the reduction ideal I_Σ, we need to verify that the S-polynomials of all ambiguities can be reduced to zero. In our example, there are 5 ambiguities, which we may informally denote by $F\underline{F}F$, $D\underline{F}F$, $\underline{K}F$, $F\underline{K}$, and $D\underline{K}$. Two of them ($F\underline{F}F$ and $D\underline{K}$) already have been dealt with above. The S-polynomials associated to the remaining ambiguities are given by

$$
\begin{aligned}
h_{DF}(\partial \otimes f) \otimes g - \partial \otimes h_{FF}(f \otimes g) &= f \otimes \partial \otimes g + (\partial f) \otimes g - \partial \otimes (fg) \\
h_{FF}(c \otimes f) - h_K(c) \otimes f &= 0 \\
h_{FF}(f \otimes c) - f \otimes h_K(c) &= 0
\end{aligned}
$$

for all $c \in M_K$ and $f, g \in M_F$. The first "family" can be reduced to zero by the rules (FF, h_{FF}) and (DF, h_{DF}) and the others are zero anyway.

Using the commands `ExtractReducibleWords`, `GenerateAmbiguities`, `SPoly`, and `ApplyRules` of the package we can verify that all S-polynomials reduce to zero in the following way.

```
ambiguities = GenerateAmbiguities[ExtractReducibleWords[RedSys]]
{Overlap[{F, F, F}, {F}, {F}],
 Overlap[{D, F, F}, {F}, {D}], SpecialInclusion[{K, F}, {}, {F}],
 SpecialInclusion[{F, K}, {F}, {}], SpecialInclusion[{D, K}, {D}, {}]}
```

```
spolys = Map[SPoly[#, RedSys] &, ambiguities]
{-Prod[F[1], mul[F[2], F[3]]] + Prod[mul[F[1], F[2]], F[3]],
 -Prod[Diff, mul[F[1], F[2]]] + Prod[Diff[F[1]], F[2]] + Prod[F[1], Diff, F[2]],
 0, 0, -Prod[Diff] + Prod[1, Diff]}
```

```
ApplyRules[spolys, RedSys]
{0, 0, 0, 0, 0}
```

This process is also available through the command `CheckResolvability`, which returns a list of all ambiguities that are not resolvable together with the reduced from of their S-polynomials, see also its application in the following section. If `CheckResolvability[RedSys]` returns the empty list, then the irreducible tensors w.r.t. the reduction system given by `RedSys` really are normal forms.

3 Applications

In this section, we illustrate how to use the package for constructing a confluent reduction system starting from a given one. For tensor reduction systems, the computer-assisted process is heuristic and users can proceed in different ways. In each step, we add new rules to the reduction system based on S-polynomials, similar to Buchberger's algorithm for computing Gröbner bases [3] and Knuth-Bendix completion [7].

As an example, we consider the algebra of integro-differential operators. Based on the free noncommutative polynomial algebra using a basis of the "function" algebra, it was introduced in [8,9] to study boundary problems; see also [10] for an automated confluence proof relying on free integro-differential algebras. First, we recall the definition of an integro-differential algebra [5,9].

Definition 1. *Let K be a commutative ring and let (R, ∂) be a differential K-algebra such that $1 \in R$ and $\partial R = R$. Moreover, let $\int\colon R \to R$ be an K-linear operation on R such that*

$$\partial \textstyle\int f = f$$

for all $f \in R$. Then we call $(R, \partial, \int)$ an integro-differential algebra *over K if the* evaluation $\mathrm{E}\colon R \to K$ *defined by* $\mathrm{E} = \mathrm{id} - \int\partial$ *is multiplicative, i.e. for all $f, g \in R$ we have* $\mathrm{E}fg = (\mathrm{E}f)\mathrm{E}g$.

We fix an integro-differential K-algebra $(R, \partial, \int)$ with ring of constants K and evaluation $\mathrm{E} = \mathrm{id} - \int\partial$. Recall from [5] that in any integro-differential algebra, we have the direct sum decomposition

$$R = K \oplus \textstyle\int R$$

into constant and non-constant "functions". We consider the corresponding K-modules $M_K = K$ and $M_{\tilde{F}} = \int R$. Note that the elements of M_K and $M_{\tilde{F}}$ are not interpreted as functions but as multiplication operators induced by those functions. For the K-linear operators ∂, $\int$, and E we consider the free modules $M_D = K\partial$, $M_I = K\int$, and $M_E = K\mathrm{E}$ generated by them. Now, let

$$M = M_F \oplus M_D \oplus M_I \oplus M_E$$

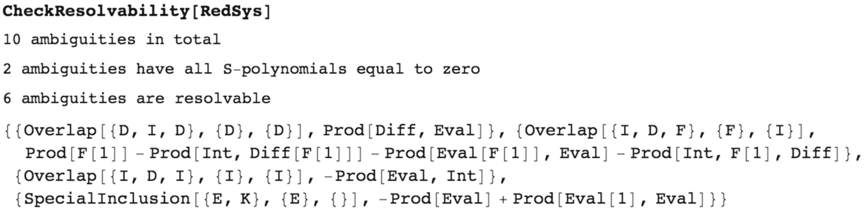
with $M_F = M_K \oplus M_{\tilde{F}}$ and alphabets $Y = \{F, D, I, E\}$ and $X = \{K, \tilde{F}, D, I, E\}$. In order to compute with these operators we need to collect the identities they satisfy in form of a reduction system. The above definition contains the following basic identities for all $f, g \in R$:

$$\partial fg = f\partial g + (\partial f)g, \quad \partial \textstyle\int g = g, \quad \int\partial g = g - \mathrm{E}g, \quad \mathrm{E}fg = (\mathrm{E}f)\mathrm{E}g.$$

Based on these, we start with the following reduction system:

$$\begin{aligned} \Sigma = \{&(FF, f \otimes g \mapsto fg), (DF, \partial \otimes f \mapsto f \otimes \partial + \partial f), (DI, \partial \otimes \textstyle\int \mapsto \epsilon), \\ &(ID, \textstyle\int \otimes \partial \mapsto \epsilon - \mathrm{E}), (EF, \mathrm{E} \otimes f \mapsto (\mathrm{E}f) \otimes \mathrm{E}), (K, 1 \mapsto \epsilon)\} \end{aligned}$$

Using our package, we determine that out of the 10 ambiguities 6 are resolvable and 4 remain. The reduced forms of the corresponding S-polynomials give rise to additional identities, among tensors as well as among elements of M_F.



```
CheckResolvability[RedSys]

10 ambiguities in total

2 ambiguities have all S-polynomials equal to zero

6 ambiguities are resolvable

{{Overlap[{D, I, D}, {D}, {D}], Prod[Diff, Eval]}, {Overlap[{I, D, F}, {F}, {I}],
  Prod[F[1]] - Prod[Int, Diff[F[1]]] - Prod[Eval[F[1]], Eval] - Prod[Int, F[1], Diff]},
 {Overlap[{I, D, I}, {I}, {I}], -Prod[Eval, Int]},
 {SpecialInclusion[{E, K}, {E}, {}], -Prod[Eval] + Prod[Eval[1], Eval]}}
```

To proceed, we introduce 3 new rules. We also update the implementation of computations in $M_F = R$ correspondingly, including the relation $\mathrm{E}1 = 1$.

$$\begin{aligned} \Sigma_1 := \Sigma \cup \{&(DE, \partial \otimes \mathrm{E} \mapsto 0), (EI, \mathrm{E} \otimes \textstyle\int \mapsto 0), \\ &\qquad (IFD, \textstyle\int \otimes f \otimes \partial \mapsto f - \int \otimes \partial f - (\mathrm{E}f)\mathrm{E})\} \end{aligned}$$

Repeating the steps above, 15 of 20 ambiguities can be resolved. Only 3 of the 5 S-polynomials not reduced to zero are essentially different, and we add corresponding rules to the reduction system.

$$\begin{aligned} \Sigma_2 := \Sigma_1 \cup \{&(EE, \mathrm{E} \otimes \mathrm{E} \mapsto \mathrm{E}), (IFE, \textstyle\int \otimes f \otimes \mathrm{E} \mapsto \int f \otimes \mathrm{E}), \\ &\qquad (IFI, \textstyle\int \otimes f \otimes \int \mapsto \int f \otimes \int - \int \otimes \int f)\} \end{aligned}$$

Now, among 37 ambiguities 35 are resolvable, resulting in two new rules.

$$\Sigma_3 := \Sigma_2 \cup \{(IE, \textstyle\int \otimes \mathrm{E} \mapsto \int 1 \otimes \mathrm{E}), (II, \int \otimes \int \mapsto \int 1 \otimes \int - \int \otimes \int 1)\}$$

Finally, we get 52 ambiguities which are all resolvable. This means that the derived reduction system Σ_3 is confluent. The function `IrreducibleWords` of the package generates all irreducible words up to a given length. One can prove that the irreducible words are given by $\tilde{F}ED^j$ and $\tilde{F}I\tilde{F}$ with $j \in \mathbb{N}_0$. Consequently, irreducible tensors are K-linear combinations of tensors of the form $f \otimes \mathrm{E} \otimes \partial^{\otimes j}$ and $f \otimes \int \otimes g$ with $j \in \mathbb{N}_0$, $f, g \in M_{\tilde{F}}$ where f, g, and E can also be absent.

References

1. Bergman, G.M.: The diamond lemma for ring theory. Adv. Math. **29**, 178–218 (1978)
2. Bokut, L.A., Chen, Y.: Gröbner-Shirshov bases and their calculation. Bull. Math. Sci. **4**, 325–395 (2014)
3. Buchberger, B.: An algorithm for finding the bases elements of the residue class ring modulo a zero dimensional polynomial ideal (German). Ph.D. thesis, University of Innsbruck (1965)
4. Gómez-Torrecillas, J.: Basic module theory over non-commutative rings with computational aspects of operator algebras. In: Barkatou, M., Cluzeau, T., Regensburger, G., Rosenkranz, M. (eds.) AADIOS 2012. LNCS, vol. 8372, pp. 23–82. Springer, Heidelberg (2014)
5. Guo, L., Regensburger, G., Rosenkranz, M.: On integro-differential algebras. J. Pure Appl. Algebra **218**, 456–473 (2014)
6. Hossein Poor, J., Raab, C.G., Regensburger, G.: Algorithmic operator algebras via normal forms for tensors. In: Proceedings of ISSAC 2016, p.8. ACM, New York (2016, to appear)
7. Knuth, D.E., Bendix, P.B.: Simple word problems in universal algebras. Computational Problems in Abstract Algebra, pp. 263–297. Pergamon, Oxford (1970)
8. Rosenkranz, M.: A new symbolic method for solving linear two-point boundary value problems on the level of operators. J. Symbolic Comput. **39**, 171–199 (2005)

9. Rosenkranz, M., Regensburger, G.: Solving and factoring boundary problems for linear ordinary differential equations in differential algebras. J. Symbolic Comput. **43**, 515–544 (2008)
10. Regensburger, G., Tec, L., Buchberger, B.: Symbolic analysis for boundary problems: from rewriting to parametrized Gröbner bases. Numerical and Symbolic Scientific Computing, pp. 273–331. Springer, Vienna (2012)