ELSEVIER Journal of Pure and Applied Algebra 117 & 118 (1997) 431–445

JOURNAL OF PURE AND APPLIED ALGEBRA

Termination for a class of algorithms for constructing algebras given by generators and relations

Marc A.A. van Leeuwen[a,*], Marcel Roelofs[b]

[a] *CWI, P.O. Box 94079, 1090 GB Amsterdam, Netherlands*

[b] *Paragon Decision Technology, P.O. Box 3277, 2001 DG Haarlem, Netherlands*

Abstract

We consider a certain type of algorithm designed to construct the multiplication table of algebras given by generators and relations. These computations may be performed for various classes of (not necessarily associative) algebras, such as Lie (super)algebras, Jordan algebras, associative algebras; in general, for any class of algebras axiomatised by suitable polynomial identities. The type of algorithm considered is based on straightforward computations in the free non-associative algebra on the generators, which do not depend in an essential way on the axioms of the class of algebras, in particular no concept of "representation" of the algebra is used. The algorithm itself is only partially specified: it proceeds by repeatedly taking steps chosen from a limited repertoire of very simple possibilities, but no fixed strategy for selecting steps is assumed. We study the question whether termination of the algorithm is guaranteed for those inputs that actually describe a finite-dimensional algebra (the question whether some arbitrary input has this property is not algorithmically decidable). We prove under certain assumptions about the strategy that termination is indeed guaranteed in this case. © 1997 Elsevier Science B.V.

1991 Math. Subj. Class.: 17-04, 17Bxx

1. Introduction

We consider some class of algebras over a field k, characterised by a set of axioms that are multivariate polynomial identities to be satisfied by all elements of the algebra. One such class is that of associative algebras, another one that of Lie algebras, which will be the running example in most of this paper. In the case of Lie algebras the product of x and y is traditionally written as $[x, y]$ to stress non-associativity of the

* Corresponding author. E-mail: m.van.leeuwen@cwi.nl.

PII S0022-4049(97)00021-2

product, and called a commutator; here the axioms are

$$[x,x]=0 \quad \text{(anti-commutativity)},$$

$$[x,[y,z]]+[z,[x,y]]+[y,[z,x]]=0 \quad \text{(Jacobi identity)},$$

for all x, y, z. In our discussion we shall fix the class of algebras to that of Lie algebras, in order to keep the formulations concrete. However, the same arguments apply to other classes of algebras as well with minor modifications: basically, whenever the Jacobi identity is mentioned, or the expression $\mathrm{Jac}(x,y,z)=[x,[y,z]]+[z,[x,y]]+[y,[z,x]]$, it should be replaced by the relevant axiom(s) or expression(s) for the class of algebras at hand. The anti-commutativity axiom of Lie algebras is treated differently, and built into the steps of the algorithm itself; while it could be treated in the same manner as the Jacobi identity is, this would be needlessly cumbersome. This serves as an example of an axiom that is sufficiently simple to be dealt with immediately, without requiring explicit processing by the algorithm; note however that in the case of associative algebras the associativity axiom cannot be handled in this way, and should be treated analogously to the Jacobi identity. The places where anti-commutativity is incorporated into our formulations will be clearly indicated, and obviously they should be altered as appropriate for other classes of algebras.

We assume that k is such that exact arithmetic with its elements is possible, e.g., $k=\mathbf{Q}$. We shall also need to make an assumption about the sets of polynomial identities figuring as axioms of our class of algebras: they should be such that it is sufficient to verify them on a k-basis of the algebra. This will certainly be the case when an expression that is linear in all its indeterminates, like $\mathrm{Jac}(x,y,z)$, is equated to 0. Axioms that do not satisfy this condition can often be made to conform by augmenting them with identities obtained from them by polarisation (substituting linear combinations of new indeterminates for old ones and simplifying, in order to obtain expressions in more indeterminates, but of lower degree in each one of them). Just as an example, consider the anti-commutativity axiom above: it is not sufficient to verify it on a k-basis. By polarisation we obtain from it the equation

$$[x,y]+[y,x]=0,$$

and verifying both identities on a k-basis is sufficient (if $\mathrm{char}\, k \neq 2$ one can even omit the original axiom).

The problem we shall deal with is that of determining a Lie algebra given by generators and relations. So let a finite set G of formal indeterminates be given (the generators), and a finite set R of relations, expressions built up from elements of G using the operations of multiplication (the Lie bracket) and formation of linear combinations with coefficients in k. We wish to find the Lie algebra specified by the generators G and the relations R, i.e., a Lie algebra L with indicated elements corresponding to the generators $g \in G$, such that the expressions obtained by substituting these elements for their corresponding generators into the relations $r \in R$ all evaluate to 0 in L, and that conversely any relation with this property can be deduced in finitely many steps from

the given relations R using the axioms of Lie algebras. To give the requested Lie algebra L will be taken to mean giving a basis of L as a vector space over k (in the form of a set of formal identifiers) while indicating the vectors that correspond to each of the generators, and giving a table expressing each product $[x, y]$ of basis elements explicitly as a k-linear combination of basis elements. Such a method of describing L is, of course, only possible if L is finite dimensional; however, we cannot expect to be able to algorithmically decide whether this is the case for given sets of generators and relations.

The purpose of this paper is not to give a detailed description of concrete algorithms used in practice to perform these computations; that has been done elsewhere for various instances of such algorithms [1, 3, 4, 8]. Rather, we study their termination properties. For termination it is necessary that the input specifies a finite-dimensional algebra; we shall establish that under certain assumptions about the strategy used by the algorithm, this condition is also sufficient. This result is analogous to what is known for the Todd–Coxeter algorithm for groups (which is however not an algorithm of the type we are considering here). There are various details of the algorithms that are important from the point of view of efficiency, but that do not affect termination. We shall abstract from such details, and simplify the algorithms to a basic form, in which only the most fundamental kinds of steps are performed. This will simplify our reasoning, but one should realise the simplified algorithm is not one that would be used in practice; yet, the termination result easily carries over to practical versions of the algorithm. The simplification also means that our reasoning applies to the various algorithms mentioned, despite their mutual differences. Note however that this paper does not apply to another type of algorithms that have been studied recently [2, 5–7], which deal with similar questions for the special cases of associative and Lie algebras, and some of which construct representations for the algebras considered (in our general setting, there is no concept of a representation of the algebra).

The algorithms we consider use the following general scheme to construct a basis and multiplication table for the desired algebra L. Starting with a basis consisting of the given set of generators G and an empty multiplication table, one proceeds to determine an increasingly large part of that table, by using available relations whenever possible, or else by adding products that cannot be determined in this way as new basis elements. The axioms serve to automatically derive relations for all basis elements of L that are introduced, which relations are treated in the same manner as those that were originally given in the set R. Therefore, the basic algebraic structure in which computations are performed is the free non-associative algebra on the basis elements currently constructed, rather than the free Lie algebra on those elements: one does not attempt to immediately impose all the consequences of the Jacobi identity on the expressions handled by the algorithm. (As mentioned above, we do take anti-commutativity directly into account, so in fact we compute in a free non-associative anti-commutative algebra.) Consequently, rewriting of expressions during the computation is very simple, and amounts to replacing any subexpressions that are products of basis elements, and for which the multiplication table under construction already

gives a value. It may happen that a linear dependency between basis elements appears during the computation, in which case one of them has to be eliminated, and the table accordingly restructured.

A concrete algorithm will incorporate a strategy for deciding in which order to extend the multiplication table, and in case of linear dependencies, for deciding which basis element to eliminate; we do not assume any one particular strategy. The assumptions about the strategy needed to obtain our termination result will be mild: essentially, one should give preference to introducing monomials of low degree as new basis elements (for some suitable rank function on the free algebra), and in case of linear dependencies, the element chosen for elimination should be of maximal degree.

2. A simple example

In order to give the flavour of the algorithms we are considering, we demonstrate a computation for a very simple case. The example is intended merely as an illustration; we do not formally define the algorithm that is being used, nor is that algorithm of the simplified form that we shall reason about below (that would make the algorithm require too many steps, even for this trivial example).

We wish to compute the Lie algebra L defined by the sets of generators $G = \{s, t\}$ and relations $R = \{[s,[s,t]] = 0,\ [t,[t,[s,t]]] = 0,\ [t,[s,t]] = [s,[t,[s,t]]]\}$ (for convenience we write the relations as equations). Noting that $[s,t]$ occurs frequently in the relations we introduce a new identifier a to stand for it, and with that replacement we similarly introduce b to stand for $[t,a]$; therefore, L is equivalently described by generators $\{s,t,a,b\}$ and relations $\{a = [s,t],\ b = [t,a],\ [s,a] = 0,\ [t,b] = 0,\ b = [s,b]\}$. Together with the anti-commutativity axiom this allows us to build a partial multiplication table (Table 1).

Now we verify the Jacobi identity for the basis elements. Since $\mathrm{Jac}(x,y,z)$ is alternating in x,y,z (any permutation of those indeterminates multiplies it by the sign of the permutation) we need only substitute triples of distinct basis elements for x,y,z, and each triple only in a fixed order. Using the partial multiplication table, $\mathrm{Jac}(s,t,b) = [s,[t,b]] + [t,[b,s]] + [b,[s,t]]$ simplifies to $[b,a]$, which allows us to complete the table by putting $[a,b] = [b,a] = 0$. We must still verify the remaining instances of the Jacobi identity however; while $\mathrm{Jac}(s,a,b)$ and $\mathrm{Jac}(t,a,b)$ now reduce to 0, we find that $\mathrm{Jac}(s,t,a) = b$, so that $b = 0$, and b is not a basis vector after all. We therefore substitute 0 for b, and remove the row and column of b from the table, and equate all

Table 1

[,]	s	t	a	b
s	0	a	0	b
t	$-a$	0	b	0
a	0	$-b$	0	
b	$-b$	0		0

Table 2

$[\cdot,\cdot]$	s	t	a
s	0	a	0
t	$-a$	0	0
a	0	0	0

entries that appeared there to 0. Since all these new equations are trivial, we complete our computation, producing multiplication Table 2 for the 3-dimensional algebra L.

3. Definitions

For a given set X we shall consider the following free structures. The free vector space $FV(X)$ over k on X consists of all formal k-linear combinations of elements of X (if X is a subset of independent vectors in some other vector space, we shall also use the notation $FV(X)$ to denote their linear span). The free magma $FM(X)$ on X is a set inductively defined as $FM(X) = X \cup (FM(X) \times FM(X))$: its elements are either elements of X or 2-tuples of elements of $FM(X)$, so that $FM(X)$ corresponds bijectively to the set of binary trees with elements of X as leaves. For an element $z \in FM(X)$ of the form $z = (x, y)$, the elements x and y are called the children of z, and z is the parent of x and of y; the reflexive transitive closures of these relations are those of descendent and ancestor. The *width* of $x \in FM(X)$ is defined to be 1 for $x \in X$, and otherwise the sum of the widths of the two children of x (this corresponds to the number of leaves of a binary tree; we avoid the term "degree" here, as it might cause confusion with the filtering defined on certain algebras below). The free algebra $FA(X)$ is equal to $FV(FM(X))$ as a vector space, and it is equipped with a (non-associative) algebra structure defined by $(x \cdot y) = (x, y)$ for $x, y \in FM(X)$, which is extended by k-bilinearity to general elements of $FA(X)$. Within $FA(X)$ the elements of $FM(X)$ are called monomials. The freeness of these structures is expressed by the fact that any map from X to a k-vector space (respectively magma, algebra) Y uniquely extends to a morphism of k-vector spaces (magmas, algebras) from $FV(X)$ ($FM(X)$, $FA(X)$) to Y.

Define a *partial multiplication table* τ on X to consist of a subset P of $X \times X$, called the domain of τ, together with a mapping $\mu_\tau : P \to FV(X)$. Since our discussion considers computations for Lie algebras, we shall write $[x, y]_\tau$ for $\mu_\tau(x, y)$, and our partial multiplication tables will be skew-symmetric: for all $x, y \in X$ one has $(x,x) \in P$ with $[x,x]_\tau = 0$, and $(x, y) \in P$ holds if and only if $(y,x) \in P$, in which case $[y,x]_\tau = -[x, y]_\tau$. For the same reason we impose anti-commutativity in the following definition. Given a set X and a skew-symmetric partial multiplication table τ on X, we define the algebra $FA(X,\tau)$ as the quotient of $FA(X)$ by the (two-sided) ideal generated by all elements of the form $(x \cdot x)$ for $x \in FA(X)$ (to enforce anti-commutativity), and all

elements of the form $(x \cdot y) - [x, y]_\tau$ for (x, y) in the domain of τ. In order to bring the monomials in $FA(X)$ into standard form with respect to anti-commutativity, we need a total ordering on $FM(X)$; such an ordering is called compatible if $x < y$ implies $(x,z) < (y,z)$ and $(z,x) < (z,y)$ for all $z \in FM(X)$.

Proposition 1. *Let τ be a skew-symmetric partial multiplication table on X, and let a compatible total ordering on $FM(X)$ be chosen. Consider the rewrite system in $FA(X)$ with reductions $z \to z'$ whenever z' is obtained by replacing a descendent (x, y) of a monomial of z in one of the following ways:* (a) *if $x = y$ it may be replaced by* 0; (b) *if $x > y$ it may be replaced by $-(y,x)$*; (c) *if $x, y \in X$ and (x, y) lies in the domain of τ, it may be replaced by $[x, y]_\tau$. In each case z' is expanded after the substitution by k-bilinearity of the product (in cases* (a) *and* (b) *this means that the indicated monomial of z is killed or negated). This rewrite system is confluent and strongly normalising.*

Recall that confluence of a rewrite system means that if a term α rewrites in two different ways to x and y, then these can be further rewritten to a common term ω; strong normalisation means that no infinite sequences of rewriting are possible. The combination of these properties obviously ensures that each term α can be rewritten to a normal form ω (a term that cannot be further rewritten), which is uniquely determined by α (Newman's lemma).

Proof. Since τ is skew-symmetric, we may save the reductions of type (a) and (b) until none of type (c) are possible; since the rewrite system consisting only of the reductions of type (a) and (b) is evidently confluent and strongly normalising, it will suffice to prove the proposition for the rewrite system with only the reductions of type (c). In such a reduction a monomial m is replaced by monomials whose width is one less than that of m, and it follows that the rewrite system is strongly normalising. Furthermore observe that if a monomial has two different descendents (x, y) and (x', y'), both in the domain of τ, then these are disjoint subtrees of width 2, from which confluence easily follows. □

Note that the essential point in the proof is the disjointness of reducible subexpressions, which is due to the non-associativity of the algebra we compute in. If we would replace $FA(X)$ by the free associative algebra on X, whose monomials are words over X rather than binary trees, then we would not have this disjointness, and the proposition would fail. In fact, because of the unsolvability of the word problem for semigroups, no strongly normalising rewrite system can determine all consequences of the relations imposed by an arbitrary partial multiplication table on an associative algebra. By contrast, the proposition shows that the implications of anti-commutativity can be incorporated into the rewrite system. It is this distinction that allows us to handle anti-commutativity immediately, whereas axioms like associativity or the Jacobi identity require the iterative approach of the algorithm presented below. This is not to

say that special properties of (free) associative or Lie algebras cannot be built into an algorithm (this is in fact what is done in [2, 5–7]), but some form of postponement of implications of the partial multiplication table is inevitable.

Denote by $N_\tau : FA(X) \to FA(X)$ the map that sends each term α to its normal form ω in the rewrite system of the proposition. Then $N_\tau^{-1}(0)$ is equal to the kernel of the canonical projection $\pi_\tau : FA(X) \to FA(X,\tau)$, so that the restriction of π_τ to the image $\mathrm{Im}(N_\tau)$ of N_τ is an isomorphism of k-vector spaces. That image is spanned by the monomials that are ordered increasingly (every descendent (x,y) has $x < y$), and that have no subtrees of width 2 lying in the domain of τ. It can serve as a model for computation in $FA(X,\tau)$, since the restriction of π_τ to $\mathrm{Im}(N_\tau)$ becomes an isomorphism of algebras when $\mathrm{Im}(N_\tau)$ is equipped with the multiplication $(x,y) \mapsto N_\tau(x \cdot y)$. We shall therefore treat $FA(X,\tau)$ as if computation with its elements and equality tests are directly possible, like for free algebras; in doing so, appropriate applications of the normal form algorithm N_τ are implicitly assumed. Since the image of $FV(X)$ in $FA(X)$ clearly lies inside $\mathrm{Im}(N_\tau)$, we shall consider it to be a subspace of $FA(X,\tau)$. Like for Lie algebras, the product in $FA(X,\tau)$ will be denoted by $[x,y]$, and so the expression $\mathrm{Jac}(x,y,z)$ is meaningful for $x,y,z \in FA(X,\tau)$; it is a non-trivial expression, since the product $[x,y]$ in $FA(X,\tau)$, though anti-commutative, does not satisfy the Jacobi identity.

4. Basic steps of the algorithm

We shall now indicate the data structures maintained by the simplified algorithms considered in this paper, and the basic operations that are performed on these data. An intermediate state of the computation will be characterised by a 4-tuple (S,m,τ,Q) where S is a finite set, m is a map from S to $FM(G)$ (recall that G is the set of generators specified in the input to the algorithm), τ is a skew-symmetric partial multiplication table on S, and Q is a finite subset of $FA(S,\tau)$. The set S consists of formal indeterminates parametrising the part of the basis of L constructed so far, and m tells how each basis element is related to the original generators (we only introduce basis elements that correspond to some product of original generators, i.e., to an element of $FM(G)$); τ is the part of the multiplication table of L that has been constructed so far, and Q is a collection of relations that still have to be processed. The initial 4-tuple is $(G, \mathrm{id}_G, \tau_0(G), R \cup \mathrm{Jac}(G,G,G))$, where $\tau_0(G)$ is the map with domain $\{(g,g) \mid g \in G\}$ and image $\{0\}$, and $\mathrm{Jac}(X,Y,Z)$ denotes the set $\{\mathrm{Jac}(x,y,z) \mid x \in X, y \in Y, z \in Z\}$. The algorithm will terminate when a state is reached in which the domain of τ is $S \times S$, and $Q = \emptyset$. We shall now consider two different kinds of steps that can be taken to transform the state into a new one; each of these steps is parametrised by values that specify the precise transformation to be performed.

Step A: *Parameters.* $s,s' \in S$ for which (s,s') does not lie in the domain of τ. Action: the 4-tuple (S,m,τ,Q) is replaced by (S',m',τ',Q'), defined as follows. Put $S' = S \cup \{c\}$ where c is an indeterminate not occurring in S; define $m'(s) = m(s)$ for all $s \in S$ and $m'(c) = (m(s), m(s'))$. The domain of τ' is that of τ united with $\{(s,s'),(s',s),(c,c)\}$;

define τ' by $[x,y]_{\tau'}=[x,y]_\tau$ for all (x,y) in the domain of τ, and furthermore $[s,s']_{\tau'}=c=-[s',s]_{\tau'}$, and $[c,c]_{\tau'}=0$. Finally put $Q'=\overline{Q}\cup \mathrm{Jac}(S,S,c)\setminus\{0\}$, where $\overline{Q}$ denotes the image of Q under the algebra morphism $FA(S,\tau)\to FA(S',\tau')$ that is the identity on S.

The indicated algebra morphism clearly exists, as every relation imposed by τ is also imposed by τ'. The effect of step A is to introduce a new indeterminate c to fill the empty slot $[s,s']$ of τ, and to substitute c for all occurrences of $[s,s']$ within Q. Because the basis is being extended, new Jacobi identities have to be taken into account as well; since $\mathrm{Jac}(x,y,z)$ is alternating with respect to permutations of x,y,z, it suffices here to use c in one fixed position only.

In itself this step does not contribute to obtaining a complete multiplication table, but its main purpose is to simplify the relations in Q by substituting indeterminates for commutators, so that eventually these relations can be used to fill in open places in the table τ. This happens when only a single commutator (monomial of width 2) is left in the relation, and the remainder is linear (monomials of width 1 only): the commutator can be singled out as left hand side of the equation, and the right-hand side gives the linear expression to fill into the corresponding position of the multiplication table. It may however happen that a relation skips this point and directly becomes completely linear; in that case it must be used to eliminate one of the elements of S, and remove it from the table. Although the latter possibility appears to be unattractive (it implies that some previous extension of the basis was unnecessary), it is impossible to rule out beforehand.

On the other hand, suppose a relation is simplified to the point that it allows a place in the table to be filled, then instead of doing that, one could choose to make it completely linear by means of one more step A, and then use it to immediately eliminate the new indeterminate; in this roundabout fashion one achieves essentially the same result as by filling the place in the table directly. For this reason we decide to simplify the description of the algorithm by omitting the obvious step of using a relation to fill in the table, while retaining the less obvious but unavoidable step of using a linear relation to eliminate a variable and reducing the table correspondingly; that will be step B below. This is a simplification only from the theoretical point of view (by reducing the number of different steps our proofs will be simpler): the actual execution of the algorithm will become more complicated by omission of the indicated kind of step, and certainly less efficient. Therefore practical implementations of this type of algorithm do not omit the step, and possibly even include other kinds of steps to efficiently handle certain situations; as long the effect of these additional steps matches that of some combination of steps A and B, our results remain valid for such improved algorithms.

Step B: Parameters. An element $r\in Q$ that lies in $FV(S)$, and some $s\in S$ that occurs in r with non-zero coefficient. Action: the 4-tuple (S,m,τ,Q) is replaced by (S',m',τ',Q'), defined as follows. Put $S'=S\setminus\{s\}$, and define $m'(s')=m(s')$ for all $s'\in S'$. Write r as $\alpha(s-v)$ with $\alpha\in k^*$ and $v\in FV(S')$. The domain of τ' is the

intersection of that of τ with $S' \times S'$, and $[x,y]_{\tau'} = \mathrm{subs}_{s:=v}([x,y]_\tau)$ for all (x,y) in the domain of τ', where the operator $\mathrm{subs}_{s:=v}$ denotes substitution of v for s. Define T as set of all expressions $[x,s]_\tau - [x,v] \in FA(S,\tau)$ with $x \in S'$ and (x,s) in the domain of τ; then $Q' = \mathrm{subs}_{s:=v}(Q \cup T) \setminus \{0\}$.

The element s is eliminated from S by substituting the linear combination v of remaining elements for it, both in the entries of the table τ and in the set of relations Q (the relation r itself, used to eliminate s, becomes 0, and is excluded from Q'). Any table entries for s should not be forgotten about, so for each entry $[x,s]_\tau$ disappearing from the table, a relation equating it the to the linear combination $[x,v]$ of products that remain inside the table is added to Q (because of anti-commutativity, we need not repeat this for the entries $[s,x]_\tau$ that also disappear from the table). Since the result is interpreted in $FA(S',\tau')$, such a relation will be linear (i.e., in $FV(S')$) if entries are defined in τ' for all those products; therefore, eliminating one linear relation may create new linear relations, allowing further reduction of the table.

5. Soundness

The algorithms we shall consider proceed by applying steps A and B to the initial data in some order, until a state is reached in which the algorithm terminates. In that state, the set S is supposed to be a basis for the desired Lie algebra L, and τ its multiplication table. Our first goal is to prove that this is indeed the case, i.e., that the definitions of the steps are correct.

Lemma 2. *Let (S',m',τ',Q') be obtained from (S,m,τ,Q) by an application of step A or of step B. Then the Lie algebra with generators S and relations $[x,y] = [x,y]_\tau$ for all (x,y) in the domain of τ, and r for all $r \in Q$, is isomorphic to the Lie algebra similarly defined for S',τ', and Q'.*

Proof. Strictly speaking, the relations $r \in Q$ should be lifted from $FA(S,\tau)$ to $FA(S)$, but since we are imposing all relations coming from τ as well, it does not matter how this is done. Any relations of the form $\mathrm{Jac}(x,y,z)$ or $[x,x]$ may be ignored, since these are automatically satisfied in a Lie algebra. Then step A amounts to introducing an extra generator c and an extra relation $[s,s'] - c$; it is obvious that the resulting Lie algebra is isomorphic to the original one, with c corresponding to $[s,s']$. Similarly, the reverse of step B amounts to introducing an extra generator s, and an extra (linear) relation $s - v$, while replacing some occurrences of v in the relations by s; here too it is obvious that an isomorphic Lie algebra is obtained. □

This lemma shows that upon termination we have a set of generators and relations (in the form of a multiplication table) that is equivalent to the ones originally given, but not that the multiplication table actually defines a Lie algebra. This point is settled by another simple lemma.

Lemma 3. *Let* (S,m,τ,Q) *be obtained from the initial state defined by* (G,R) *by steps of type A and B, then* $\mathrm{Jac}(S,S,S)\subseteq \pm Q\cup\{0\}$.

Proof. By induction on the number of steps taken. In step A all additional expressions $\mathrm{Jac}(x,y,z)$ in $\mathrm{Jac}(S',S',S')$ are (up to a sign) explicitly included in Q', while the ones already in Q either have a counterpart in Q', or become 0 in $FA(S',\tau')$. For step B the situation is even simpler, since all elements of $\mathrm{Jac}(S',S',S')$ are directly obtained from elements of $\mathrm{Jac}(S,S,S)$. □

Theorem 4. *If from an initial state* $(G,\mathrm{id}_G,\tau_0(G),R\cup\mathrm{Jac}(G,G,G))$ *a state* $(S,m,\tau,\emptyset)$ *is reached by a series of steps A and B, with the domain of* τ *equal to* $S\times S$, *then the Lie algebra defined by generators G and relations R is isomorphic to the algebra defined on the space* $FV(S)$ *by the multiplication table* τ. □

6. Termination

We shall now consider the question whether the fact that the Lie algebra defined by the generators G and relations R is finite dimensional is sufficient to guarantee termination of the algorithm for those input data. This will depend on the strategy for choosing between steps A and B and, more importantly, choosing the parameters for these steps. In this section we shall consider an arbitrary algorithm that successively applies instances of steps A and B according to some strategy, and shall make a sequence of assumptions about the strategy; when all assumptions are satisfied, we shall answer the above question affirmatively. Each statement that appears will apply to algorithms whose strategy satisfies all the assumptions stated up to that point, without explicitly mentioning so each time.

Consider at each stage of the algorithm the map $p_\tau: FA(G)\to FA(S,\tau)$ describing how expressions in the original generators would be rewritten, if they were to undergo the sequence of transformations that that are applied to the elements of Q in the preceding steps (we shall use a subscript τ for quantities that depend on the state of the computation, rather than the full state (S,m,τ,Q)). For reasoning about the algorithm it is important that p_τ be an algebra morphism; this is not necessarily always the case, however. The problem is that the maps $FA(S,\tau)\to FA(S',\tau')$ describing the rewriting for each individual step are not always algebra morphisms; in particular, this fails for of step B if the set $\mathrm{subs}_{s:=v}(T)$ of new relations added to Q' contains non-zero elements. If $r=\mathrm{subs}_{s:=v}([x,s]_\tau-[x,v])$ is such a relation, then any element $e\in FA(G)$ with $p_\tau(e)=[x,s-v]=[x,s]_\tau-[x,v]$ will have $p_{\tau'}(e)=r\neq 0$, but of course $p_{\tau'}(e')=0$ for any $e'\in p_\tau^{-1}(s-v)$. The difficulty is of transient nature, because when eventually r is removed from Q by another step B, then e will be rewritten to 0, as it should. Moreover, if all the relations r of this kind produced by the original step B are linear, then we can perform the steps B to eliminate these relations immediately after that first step, and then any similarly further implied steps B; if this entire sequence of steps B

starts in state (S,m,τ,Q), and ends in a state (S'',m'',τ'',Q'') in which no further steps B are implied, and then there will be an algebra morphism $FA(S,\tau)\to FA(S'',\tau'')$.

Assumption 1. For any step B that transforms the state (S,m,τ,Q) into (S',m',τ',Q'), one has $\mathrm{subs}_{s:=v}(T)\subseteq FV(S')$ for the set $T\subseteq FA(S,\tau)$ of new relations added to Q; also, for any non-zero $r\in\mathrm{subs}_{s:=v}(T)$, the step B with parameter r that could be taken immediately afterwards, recursively satisfies the same requirement. Moreover, such newly introduced relations $r\in Q'$ (and any relations obtained later by rewriting them) are marked as urgent; while Q contains urgent relations, no step is taken that is not a step B with parameter r that is urgent.

This assumption may appear to be hard to live up to in practice, but the situation is not so bad. First of all, we have seen that to get $T\subseteq FV(S')$ for a step B that eliminates s by substituting $v\in FV(S')$ for it, it suffices that whenever (x,s) is in the domain of τ, then so is (x,s') for any basis element s' occurring in v. The latter may be achieved by first performing steps A for any such pairs (x,s') that are not yet in the domain of τ; alternatively, some strategies may avoid the problem altogether by the order in which they fill entries in the table (step A), and their selection of variables to eliminate in step B. Secondly, the assumption is certainly not a necessary condition for ensuring termination of the algorithm for finite-dimensional algebras. Indeed it can be seen that finitely many interchanges of steps will not affect termination of the algorithm, so in practice there is no objection to postponing the steps A indicated above until after the initial step B, to linearise a relation that was not linear when it was first added to Q. Similarly, it is not really a crime to perform a non-urgent step while urgent steps are still possible, as long as the latter will still be taken eventually. We do need to make the assumption here however, in order to have sufficiently many algebra morphisms p_τ available; once termination is proved for strategies that satisfy the assumption exactly, it will follow for other strategies that are close enough to them.

Proposition 5. *For a state (S,m,τ,Q), if Q contains no urgent relations, p_τ is an algebra morphism.*

Proof. Call a state *transient* if Q contains urgent relations, and *stable* otherwise. Our assumption ensures the existence of an algebra morphism $FA(S,\tau)\to FA(S',\tau')$ for every pair of successive stable states, which is given for step A by the identity on S, and for a sequence of steps B by the corresponding sequence of substitutions $\mathrm{subs}_{s:=v}$: by construction every relation given by an entry of τ also holds when the elements of S are replaced by their images. The map $FA(S,\tau)\to FA(S',\tau')$ describing rewriting clearly coincides with this morphism, and by definition p_τ is the composition of such rewriting maps. □

In the sequel, whenever we mention p_τ, we shall implicitly assume that that corresponding state is stable; since any transient state is followed after finitely many steps B

by a stable state, this will not affect the validity of our argument. For stable states we define $L_\tau = p_\tau^{-1}(FV(S))$, the subspace of $FA(G)$ of expressions that would be rewritten to linear ones in $FA(S,\tau)$, and $K_\tau = p_\tau^{-1}(0)$, the ideal of expressions that would be rewritten to 0. Since an expression that is rewritten to a linear one or to 0 will subsequently remain linear respectively 0, these sets are weakly increasing as the computation progresses. Now a necessary condition to guarantee termination is that the strategy be such that the union of all L_τ, taken over a possibly infinite computation, is all of $FA(G)$; without this property there could be relations $r \in Q$ that never become eligible as parameter for step B because they fail to become linear, and yet they might by essential for obtaining a finite-dimensional algebra. The following will be useful to ensure such a property.

Assumption 2. A rank function rk: $FM(G) \to \mathbf{N}$ is defined such that only finitely many monomials of any given rank exist, and such that $\mathrm{rk}(x,y) = \mathrm{rk}(y,x)$ and $\mathrm{rk}\,x \leq \mathrm{rk}\,y \Rightarrow \mathrm{rk}(x,z) \leq \mathrm{rk}(y,z)$.

Examples of such rank functions are taking the width of monomials (the number of leaves when viewed as a tree), or the maximal nesting level (depth of the tree). The precise nature of the rank function is not very important; one need not even have $\mathrm{rk}\,g = 1$ for all $g \in G$.

Proposition 6. *If $m,n \in FM(G)$ are such that m is a proper descendent of n, then $\mathrm{rk}\,m < \mathrm{rk}\,n$.*

Proof. Assume to the contrary that $\mathrm{rk}\,m \geq \mathrm{rk}\,n$. Then consider the infinite sequence of monomials $m_0 = m$, $m_1 = n$, m_2, m_3 ..., where m_{i+1} is obtained from n by substituting m_i for the subterm m. Then clearly all m_i are distinct, and by induction one has $\mathrm{rk}\,m_i \geq \mathrm{rk}\,m_{i+1}$, which contradicts the fact that only finitely many monomials can have the same rank. □

The rank function defines a filtering of $FA(G)$ by finite-dimensional subspaces $FA(G)_i$, spanned by $\{x \in FM(G) \mid \mathrm{rk}\,x \leq i\}$. There is an induced rank function on the basis of $FA(S,\tau)$ represented by $FM(S) \cap \mathrm{Im}(N_\tau)$ (the monomials that are in normal form for N_τ): for such a monomial x define $\mathrm{rk}\,x = \mathrm{rk}\,x'$, where $x' \in FM(G)$ is obtained by substituting $m(s)$ for s in x for every $s \in S$. Again this defines a filtering of $FA(S,\tau)$ by subspaces $FA(S,\tau)_i$, spanned by $\{x \in FM(S) \cap \mathrm{Im}(N_\tau) \mid \mathrm{rk}\,x \leq i\}$.

Assumption 3. Whenever step B is taken, with as parameters a relation r and a basis element s that occurs in r with non-zero coefficient, s has maximal rank among such basis elements.

Proposition 7. *One has $p_\tau(FA(G)_i) \subseteq FA(S,\tau)_i$ for all $i \in \mathbf{N}$.*

Proof. It suffices to show that for each step, rewriting of elements maps each $FA(S,\tau)_i$ into $FA(S',\tau')_i$. For step A this is certainly the case, since $m'(c)=(m(s),m(s'))$ when $[s,s']_{\tau'}=c$. For step B, rewriting consists of substituting for $s\in S$ the linear combination $v\in FV(S')$; because s was chosen with maximal rank by Assumption 3, we have $v\in FA(S',\tau')_{\mathrm{rk}\,s}$, and due to the properties of the rank function, this implies that we have $\mathrm{subs}_{s:=v}(x)\in FA(S',\tau')_{\mathrm{rk}\,x}$ for every monomial $x\in FM(S)\cap \mathrm{Im}(N_\tau)$. □

Now let $r\in\mathbf{N}$ be such that for all $s,s'\in S$ with $[s,s']\in FA(S,\tau)_r$, the pair (s,s') lies in the domain of τ. Then by Proposition 6 we have $FA(S,\tau)_r\subseteq FV(S)$, and by Proposition 7 also $p_\tau(FA(G)_r)\subseteq FV(S)$. Consequently, for the union of all L_τ to be all of $FA(G)$, it suffices that for arbitrarily large r the stated condition will eventually be satisfied (i.e., for some τ). This can be achieved by selecting in step A pairs (s,s') with minimal possible value of $\mathrm{rk}([s,s'])$, or at least ensuring that all such elements are chosen within a finite number of steps.

Assumption 4. While there exist eligible pairs (s,s') for step A that have $\mathrm{rk}([s,s'])=i$, the strategy will not infinitely often select step A for parameters (t,t') with $\mathrm{rk}([t,t'])>i$ instead.

Lemma 8. $\bigcup_\tau L_\tau = FA(G)$. □

Assumption 5. While a relation $r\in Q$ eligible for step B exists, the strategy will not infinitely often select other steps instead.

Corollary 9. $R\cup \mathrm{Jac}(FM(G),FM(G),FM(G))\subseteq\bigcup_\tau K_\tau$.

Proof. Let $r\in R\cup\mathrm{Jac}(FM(G),FM(G),FM(G))$, then $r\in L_\tau$ for some τ, by Lemma 8. If $r=\mathrm{Jac}(x,y,z)$, then also $x,y,z\in L_\tau$, so by Lemma 3, $p_\tau(r)$ lies in the k-subspace of $FA(S,\tau)$ generated by $Q\cap FV(S)$; for $r\in R$ this holds as well. Therefore, $p_\tau(r)$ is a linear combination of elements of Q that are eligible for step B; by Assumption 5, these will eventually all be selected. Then, for the state (S',m',τ',Q') obtained after the last of these relations is selected for step B, we have $r\in K_{\tau'}$. □

This establishes that the ideal $\bigcup_\tau K_\tau$ is the kernel of the algebra morphism p: $FA(G)\to L$, where L is the Lie algebra defined by the generators G and relations R.

Theorem 10. *If the Lie algebra L defined by (G,R) is finite dimensional, and the strategy satisfies the given assumptions, the algorithm terminates.*

Proof. Since $p: FA(G)\to L$ is surjective and L is finite dimensional, there is some $d\in N$ for which the restriction of p to $FA(G)_d$ is already surjective; we may assume that moreover $G\subseteq FA(G)_d$. The set $\{\,m\in FM(G)\mid \mathrm{rk}\,m\le d\,\}$ is finite, so we may write it as $\{m_1,\ldots,m_n\}$; by the assumed surjectivity, there exist coefficients $c_k^{i,j}$ for $1\le i,j,k\le n$ such that $[p(m_i),p(m_j)]=\sum_k c_k^{i,j}p(m_k)$. Then the elements

$r_{i,j} = [m_i, m_j] - \sum_k c_k^{i,j} m_k$ lie in the kernel of p, which is the ideal generated by $R \cup \mathrm{Jac}(FM(G), FM(G), FM(G))$. As there are only finitely many $r_{i,j}$, they already lie in the ideal generated by some finite subset F of $R \cup \mathrm{Jac}(FM(G), FM(G), FM(G))$. From Lemma 8 and Corollary 9 it follows that the computation will reach a state for which both $FA(G)_d \subseteq L_\tau$ and $F \subseteq K_\tau$, which implies $p_\tau(FA(G)_d) \subseteq FV(S)$ and $p_\tau(r_{i,j}) = 0$ for all i, j. Now $p_\tau(FA(G)_d) \subseteq FA(S, \tau)_d$ by Proposition 7, so every $s \in S$ occurring with non-zero coefficient in some element of $p_\tau(FA(G)_d)$ has $\mathrm{rk}\, s \leq d$, so that $s \in p_\tau(FA(G)_d)$; therefore $p_\tau(FA(G)_d) = FV(\Sigma)$ for some $\Sigma \subseteq S$. From the vanishing of $p_\tau(r_{i,j})$ it follows that $\Sigma \times \Sigma$ is contained in the domain of τ, and that $FV(\Sigma)$ is a subalgebra of $FA(S, \tau)$; since this subalgebra contains $p_\tau(G)$ is must be equal to all of $FA(S, \tau)$, and $\Sigma = S$. Then τ is a complete multiplication table, and no further steps A are possible; the algorithm will therefore terminate, after possibly taking a finite number of steps B to obtain $Q = \emptyset$. □

7. Other classes of algebras

To conclude, we summarise the points that should be altered in the steps of the algorithm and the proofs above to adapt them to classes of algebras other than Lie algebras. As stated earlier, we assume that such a class of algebras is axiomatised by a set of polynomial identities with the property that if each instance of these identities obtained by substituting elements from a given k-basis for all variables is satisfied, then all instances obtained by substituting arbitrary vectors for the variables are also satisfied. Any axiom that, like anti-commutativity, is such that the corresponding variation of Proposition 1 is valid, can be given a similar special treatment as we gave to anti-commutativity; however, this is never necessary, and we shall assume that all axioms are handled similarly to the Jacobi identity.

The definition of $FA(X, \tau)$ is modified by removing the assumption that τ is skew-symmetric, and the quotient is taken for the ideal generated by elements of the form $(x \cdot y) - \mu_\tau(x, y)$ only. Proposition 1 is simplified by omitting the reduction steps of types (a) and (b), and a compatible total ordering on $FM(X)$ is no longer required. The algorithm itself is modified as follows: the data are the same, except that again τ is no longer assumed to be skew-symmetric; the initial table τ_0 is completely empty; for the initial value of Q we adjoin to R all instances of all axioms obtained by substituting elements of G for the variables; in step A only $[s, s']_{\tau'} = c$ is defined in addition to the values copied from τ, and the set of identities joined to $\overline{Q}$ consists of all instances of all axioms with c substituted at least once for a variable, and elements of S substituted for the remaining variables; in step B the set T consists of all expressions $\mu_\tau(x, s) - (x \cdot v)$ and $\mu_\tau(s, x) - (v \cdot x)$ with $x \in S$ and (x, s), respectively, (s, x) in the domain of τ. The soundness proof remains valid, with the term Lie algebra replaced by the class of algebras under consideration; in Lemma 3 the '$\pm$' may even be omitted. The termination proof can be similarly retained, replacing $\mathrm{Jac}(FM(G), FM(G), FM(G))$ by the set of all instances of all axioms obtained by substituting elements of $FM(G)$

for the variables; the requirement $\mathrm{rk}(x, y) = \mathrm{rk}(y, x)$ for the rank function is no longer needed.

References

[1] I.R. Akselrod, V.P. Gerdt, V.E. Kovtun and V.N. Robuk, Construction of a Lie algebra by a subset of generators and commutation relations, in: D.V. Shirkov, V.A. Rostovsev, V.P. Gerdt, eds., Computer Algebra in Physical Research (World Scientific, Singapore, 1991) 306–312.

[2] V.P. Gerdt and V.V. Kornyak, Construction of finitely presented Lie algebras and superalgebras, preprint JINR, Dubna (1995) To appear in: J. Symbol. Comput.

[3] V.P. Gerdt and V.N. Robuk, V.M. Severyanov, On construction of finitely presented Lie algebras, preprint JINR E5-94-302, Dubna (1994) Submitted to: Comput. Maths and Math. Phys.

[4] P.K.H. Gragert, Lie algebra computations, Acta Appl. Math. 16 (1989) 231–242.

[5] G. Labonté, An algorithm for the construction of matrix representations for finitely presented non-commutative algebras, J. Symbol. Comput. 9 (1990) 27–38.

[6] S.A. Linton, Constructing matrix representations of finitely presented groups, J. Symb. Comput. 12 (1991) 427–438.

[7] S.A. Linton, On vector enumeration, Linear Alg. Appl. 192 (1993) 235–248.

[8] G.H.M. Roelofs, The LIESUPER package for REDUCE, Memorandum 943, Univ. of Twente, The Netherlands (1991).