Labeled Trees and the Algebra of Differential Operators

Robert Grossman *
University of California, Berkeley

Richard G. Larson†
University of Illinois at Chicago

1988

This is a draft of a paper which later appeared in Graphs and Algorithms, Contemporary Mathematics Volume 89, B. Richter, editor, American Mathematical Society, Providence, 1989, pages 81-87.

1 Introduction

This paper is concerned with the effective symbolic computation of operators under composition. Examples include differential operators under composition and vector fields under the Lie bracket. A basic fact about such operators is that, in general, they do not commute. A basic fact about applied mathematicians is that they often rewrite expressions involving noncommuting operators in terms of other operators which do commute. If the original expression enjoys a certain symmetry, then the naive rewriting requires the computation of terms which in the end cancel.

In this paper we analyse data structures consisting of formal linear combinations of rooted labeled trees. We define a multiplication on rooted labeled trees, thereby making the set of these data structures into an associative algebra. We then define an algebra homomorphism from the original algebra of operators into this algebra of trees. The cancellation which occurs

*The first author is a National Science Foundation Postdoctoral Research Fellow.

†This paper was written while the second author was on sabbatical leave at the University of California, Berkeley.

when noncommuting operators are expressed in terms of commuting ones occurs naturally when the operators are represented using this data structure. This leads to an algorithm which, for operators which are derivations, speeds up the computation exponentially in the degree of the operator.

We first consider a concrete example. Fix three vector fields E_1, E_2, E_3 in $\mathbf{R}^N$ with polynomial coefficients a_i^j:

$$E_i = \sum_{j=1}^{N} a_i^j \frac{\partial}{\partial x_j}, \qquad \text{for } i = 1,\ 2,\ 3.$$

Considering the vector fields as first-order differential operators, it is natural to form higher-order differential operators from them, such as the third-order differential operator

$$p = E_3E_2E_1 - E_3E_1E_2 - E_2E_1E_3 + E_1E_2E_3.$$

Writing this differential operator in terms of the $\partial/\partial x_1, \ldots, \partial/\partial x_N$ yields a first-order differential operator because of the symmetry of the expression p causes all second- and third-order terms to cancel.

In this paper we analyse an algorithm for computing differential operators $L^{\#}$ which does not involve the explicit computation of the second- and third-order terms which cancel. However, there is some overhead: if $L^{\#}$ involves some cancellation there will be overall savings; otherwise, the computation of $L^{\#}$ will be slightly more expensive. In the example above, the naive computation would require the computation of $24N^3$ terms, while the algorithm we describe here would involve just the computation of the $6N^3$ terms which do not cancel.

We conclude this introduction with some remarks.

1. In actual applications expressions possessing symmetry arise more often than not. For example, Lie brackets of vector fields possess a great deal of symmetry, as does the Laplacian

   $$L = E_1E_1 + E_2E_2 + E_3E_3$$

   built from the vector fields. The algorithm we discuss is designed to take advantage of such symmetries, if they are present, without the necessity of explicitly identifying the symmetries.

2. Once a set of data structures has been given an algebraic structure, it becomes natural to view algorithms concerned with simplification

as simply factoring a map through the algebra of these data structures. This is the simple idea which is at the basis of the algorithm we describe. We expect that this idea will find application elsewhere.

3. The space of operators on a linear space is not only an algebra but also a coalgebra; that is, it is the dual of an algebra. The algebra of data structures mentioned above also has a coalgebra structure. Although this fact plays a relatively minor role in the simple algorithms discussed in this paper, it does play a crucial role for other algorithms we have studied.

4. The algorithm described here lends itself to parallelization which we are currently investigating.

5. See [3] and [2] for previous work on the simplification of expressions. The collection [1] contains several articles and many references pertaining to symbolic computation; in particular, the use of derivations in symbolic computations occurs in integration algorithms. For unexplained notions concerning data structures, see [8].

2 Higher-order derivations

In this section we give a careful statement of the problem, and state the main result. Let R be a commutative algebra with unit over the field k. (Throughout this paper k is a field of characteristic 0.) A *derivation* of the algebra R is a linear map D of R to itself satisfying

$$D(ab) = aD(b) + bD(a), \qquad \text{for all } a, b \in R.$$

Let $D_1, \ldots, D_N$ be N commuting derivations of R, that is, for $i, j = 1, \ldots, N$,

$$D_i D_j a = D_j D_i a, \qquad \text{for all } a \in R.$$

Suppose that we are also given M derivations $E_1, \ldots, E_M$ of R which can be expressed as R-linear combinations of the derivations D_i; that is, for $j = 1, \ldots, M$,

$$E_j = \sum_{\mu=1}^{N} a_j^\mu D_\mu, \qquad \text{where } a_j^\mu \in R. \tag{1}$$

We are interested in writing higher-order derivations generated by the $E_1, \ldots, E_M$ in terms of the commuting derivations $D_1, \ldots, D_N$. More formally, let $k<E_1, \ldots, E_M>$ denote the free associative algebra in the symbols E_1,

$\ldots$, E_M and let $\mathbf{Diff}(D_1, \ldots, D_N; R)$ denote the space of formal linear differential operators with coefficients from R; that is, $\mathbf{Diff}(D_1, \ldots, D_N; R)$ consists of all finite formal expressions

$$L = \sum_{\mu_1=1}^{N} a_{\mu_1} D_{\mu_1} + \sum_{\mu_1,\mu_2=1}^{N} a_{\mu_1\mu_2} D_{\mu_2} D_{\mu_1} + \cdots$$

where $a_{\mu_1}, a_{\mu_1\mu_2}, \ldots \in R$. We let

$$\chi : k{<}E_1, \ldots, E_M{>} \to \mathbf{Diff}(D_1, \ldots, D_N; R)$$

denote the map which sends $p \in k{<}E_1, \ldots, E_M{>}$ to the linear differential operator $L^{\#} = \chi(p)$ obtained by performing the substitution (1), and simplifying using the fact that the D_μ are derivations of R.

Suppose $p \in k{<}E_1, \ldots, E_M{>}$ is of the form

$$p = \sum_{i=1}^{l} p_i,$$

where each term p_i is of degree m. The naive computation of $L^{\#} = \chi(p)$ would compute $\chi(p_i)$, for $i = 1, \ldots, l$. This would yield $l\, m!\, N^m$ terms. Assume $\mathrm{Cost}_{\mathrm{A}}(p)$, the cost of applying algorithm A to simplify $p \in k{<}E_1, \ldots, E_M{>}$, is proportional to the number of differentiations and multiplications. Then

$$\mathrm{Cost}_{\mathrm{NAIVE}}(p) = O(lm\, m!\, N^m)$$

In this paper we analyze an algorithm that preprocesses an expression p in such a way that any terms which cancel after the subtitution (1) are not computed. In the final section we show

Theorem 1 *Assume*

1. *p is the sum of $l = 2^{m-1}$ terms, each homogenous of degree m;*
2. *$L = \chi(p)$ is a linear differential operator of degree 1.*

Then

$$\frac{\mathrm{Cost}_{\mathrm{BETTER}}(p)}{\mathrm{Cost}_{\mathrm{NAIVE}}(p)} = O(2^{-m}).$$

Observe that a Lie bracket of degree m satisfies the hypotheses of the theorem.

3 Hopf algebras

In the next section we will give a Hopf algebra structure on trees and differential operators. In this section we summarize the definitions and needed properties of Hopf algebras.

If A is an associative algebra with unit over the field k, we can describe the structure of A with the linear maps $\mu : A \otimes A \to A$ defined by $\mu(a \otimes b) = ab$ for $a, b \in A$, and $\eta : k \to A$ defined by $\eta(x) = x1$ for $x \in k$. The fact that multiplication is associative can be restated as $\mu \circ (I \otimes \mu) = \mu \circ (\mu \otimes I)$, where $I : A \to A$ denotes the identity map. The fact that 1 is the multiplicative unit can be restated as $\mu \circ (\eta \otimes I) = \mu \circ (I \otimes \eta) = I$, where we identify A with $k \otimes A$ and $A \otimes k$ via the canonical isomorphisms. The dual notion to an algebra is a coalgebra:

Definition 2 *A* coalgebra *over the field k is a vector space C over k, together with maps $\Delta : C \to C \otimes C$ and $\epsilon : C \to k$ such that $(\Delta \otimes I) \circ \Delta = (I \otimes \Delta) \circ \Delta$ and $(\epsilon \otimes I) \circ \Delta = (I \otimes \epsilon) \circ \Delta = I$. The map Δ is called the* comultiplication *of C, and the map ϵ is called the* counit *of C.*

The coalgebra C is called cocommutative *if $\Delta = T \circ \Delta$ where $T : C \otimes C \to C \otimes C$ is defined by $T(a \otimes b) = b \otimes a$.*

If A is a finite dimensional algebra with multiplication μ and unit η, then the linear dual A^* is a coalgebra, with comultiplication μ^* and counit η^*. A Hopf algebra is both an algebra and a coalgebra, together with additional structure:

Definition 3 *A* Hopf algebra *is a vector space A over the field k, together with maps $\mu : A \otimes A \to A$, $\eta : k \to A$, $\Delta : A \to A \otimes A$, and $\epsilon : A \to k$, such that*

1. *μ and η define an algebra structure on A;*
2. *Δ and ϵ define a coalgebra structure on A;*
3. *the maps Δ and ϵ are algebra homomorphisms.*

If L is a Lie algebra (see [5] for the definition and basic properties) it can be shown that the universal enveloping algebra $U(L)$ is a Hopf algebra. Comultiplication is defined by $\Delta(x) = 1 \otimes x + x \otimes 1$ for $x \in L$, and extended to $U(L)$ by using the facts that L generates $U(L)$ as an algebra and that Δ is an algebra homomorphism. The counit is defined by $\epsilon(x) = 0$ for $x \in L$.

A vector space V over k is said to be *graded* if it is the direct sum of a family of subspaces indexed by the natural numbers:

$$V = \bigoplus_{n \geq 0} V_n.$$

A graded vector space V is said to be *connected* if $V_0 \cong k$. A graded vector space V is said to be *positively graded* if $V_0 = 0$. The tensor product of two graded vector spaces is graded as follows:

$$(V \otimes W)_n = \sum_{p+q=n} V_p \otimes W_q.$$

An algebra (coalgebra, Hopf algebra, Lie algebra, ...) is graded if it is a graded vector space, and if all of the structure-defining maps send the n-th part of their domain into the n-th part of their range. It can be shown that if L is a positively graded Lie algebra, then $U(L)$ is a graded connected Hopf algebra.

Definition 4 *Let A be a Hopf algebra.*

$$P(A) = \{\, a \in A \mid \Delta(a) = 1 \otimes a + a \otimes 1 \,\}.$$

If A is a Hopf algebra, it can be proved that $P(A)$ is a sub-Lie algebra of A^-. If L is a Lie algebra, it can be proved that $L = P(U(L))$. A partial converse to this fact plays a key role in the structure of Hopf algebras:

Theorem 5 (Milnor–Moore) *Let A be a graded connected Hopf algebra. Then $A \cong U(P(A))$ as Hopf algebras.*

See [6] or [7] for a proof of this.

4 Trees and Hopf algebras

In this section we describe the connection between Hopf algebras and trees which is essential for the description of the data structures which we use in the next section, and for the analysis of the algorithms which use those data structures.

By a tree we mean a rooted finite tree [8]. If $\{E_1, \ldots, E_M\}$ is a set of symbols, we will say a tree is *labeled with* $\{E_1, \ldots, E_M\}$ if every node of the tree other than the root has an element of $\{E_1, \ldots, E_M\}$ assigned to it. We denote the set of all trees labeled with $\{E_1, \ldots, E_M\}$ by $LT(E_1, \ldots, E_M)$.

Let $\mathcal{O} = \mathcal{O}_k\left(LT(E_1, \ldots, E_M)\right)$ denote the vector space over k with basis $LT(E_1, \ldots, E_M)$. We show that this vector space is a graded connected cocommutative Hopf algebra, and describe $P(\mathcal{O})$.

We define the multiplication in $\mathcal{O}$ as follows. Since the set of labeled trees form a basis for $\mathcal{O}$, it is sufficient to describe the product of two labeled trees. Suppose t_1 and t_2 are two labeled trees. Let $s_1, \ldots, s_r$ be the children of the root of t_1. If t_2 has $n+1$ nodes (counting the root), there are $(n+1)^r$ ways to attach the r subtrees of t_1 which have $s_1, \ldots, s_r$ as roots to the labeled tree t_2 by making each s_i the child of some node of t_2, keeping the original labels. The product $t_1 t_2$ is defined to be the sum of these $(n+1)^r$ labeled trees. It can be shown that this product is associative, and that the tree consisting only of the root is a multiplicative identity.

We define the comultiplication on $\mathcal{O}$ as follows. Let t be a labeled tree, and let $s_1, \ldots, s_r$ be the children of the root of t. If P is a subset of $C_t = \{s_1, \ldots, s_r\}$, let t_P be the labeled tree formed by making the elements of P the children of a new root, keeping the original labels. Define $\Delta(t) = \sum_{P \subseteq C_t} t_P \otimes t_{C_t \backslash P}$, where $X \backslash Y$ denotes the set-theoretic relative complement of Y in X. Define $\epsilon(t)$ to be 1 if t has only one node (its root), and 0 otherwise. It can be shown that this makes $\mathcal{O}$ into a cocommutative coalgebra. We can define a grading on $\mathcal{O}$ by letting $\mathcal{O}_n$ be the subspace of $\mathcal{O}$ spanned by the trees with $n+1$ nodes. The following theorem can be proved:

Theorem 6 *$\mathcal{O}_k(LT(E_1, \ldots, E_M))$ is a cocommutative graded connected Hopf algebra.*

The Milnor–Moore Theorem now says that we will know the structure of $\mathcal{O}$ once we know $P(\mathcal{O})$.

Theorem 7 *The set of labeled trees t whose root has exactly one child is a basis for $P(\mathcal{O})$.*

Proof: It is immediate that any tree whose root has only one child is primitive. To show that these trees span the primitive elements, define a linear map $\pi : \mathcal{O} \otimes \mathcal{O} \to \mathcal{O}$ as follows: if t_1 and t_2 are labeled trees, let $\pi(t_1 \otimes t_2)$ be the labeled tree formed by identifying the roots of t_1 and t_2. In other words, $\pi(t_1 \otimes t_2)$ is the labeled tree which has as subtrees of the root all the subtrees of the roots of t_1 and t_2. It is easy to see that if t is a labeled tree whose root has r children, then $\pi \circ \Delta(t) = 2^r t$. On the other hand, if $a = \sum a_t t \in P(\mathcal{O})$, we have that $\pi \circ \Delta(a) = 2a$. Since the trees t are linearly independent, it follows that $a_t = 0$ if the root of t has more than one child.

If $\{E_1, \ldots, E_M\}$ is a set of symbols, then the free associative algebra $k<E_1, \ldots, E_M>$ is a graded connected cocommutative Hopf algebra, and there is a Hopf algebra homomorphism

$$\phi : k<E_1, \ldots, E_M> \to \mathcal{O}_k\left(LT(E_1, \ldots, E_M)\right).$$

The map ϕ sends E_i to the labeled tree with two nodes: the root, and a child of the root labeled with E_i; it is then extended to all of $k<E_1, \ldots, E_M>$ by using the fact that it is an algebra homomorphism.

5 Simplification of higher order derivations

In this section we define a map

$$\psi : \mathcal{O}_k(LT(E_1, \ldots, E_M)) \to \mathbf{Diff}(D_1, \ldots, D_N; R).$$

We do this in several steps.

Step 1. Given a labeled tree $t \in LT_m(E_1, \ldots, E_M)$, assign the root the number 0 and assign the remaining nodes the numbers $1, \ldots, m$. From now on we identify the node with the number assigned to it. To each node k we asociate a summation index μ_k. Write $\mu = (\mu_1, \ldots, \mu_m)$.

Step 2. For $t \in LT_m(E_1, \ldots, E_M)$, let k be a node of t, labeled with E_{γ_k}, and suppose that $l, \ldots, l'$ are the children of k. Let

$$\begin{aligned} R(k;\mu) &= D_{\mu_l} \cdots D_{\mu_{l'}} a^{\mu_k}_{\gamma_k} \qquad \text{if } k \text{ is not the root;} \\ &= D_{\mu_l} \cdots D_{\mu_{l'}} \qquad \text{if } k \text{ is the root.} \end{aligned}$$

Note that if $k > 0$, then $R(k;\mu) \in R$.

Step 3. Define

$$\psi(t) = \sum_{\mu_1, \ldots, \mu_m = 1}^{N} R(m;\mu) \cdots R(1;\mu) R(0;\mu).$$

Step 4. Extend ψ to all of $\mathcal{O}_k(LT(E_1, \ldots, E_M))$ by linearity.

The next two propositions describe fundamental properties of the map ψ. Note that the next proposition is an example of simplification by factoring χ through the set of labeled trees: we will see that often ψ and ϕ together are cheaper to compute than χ.

Proposition 8

$$\chi = \psi \circ \phi.$$

The proof is a straightforward verification and is contained in [4]. In fact more is true: the map ψ respects the interaction of the comultiplication on $\mathcal{O}_k(LT(E_1, \ldots, E_M))$ and the multiplication of R in the following sense:

Proposition 9 *For all $a, b \in R$, and for all $t \in \mathcal{O}_k(LT(E_1, \ldots, E_M))$,*

$$((\psi \otimes \psi) \circ \Delta(t))\,(a \otimes b) = \psi(t)(ab).$$

We now consider the cost of writing an expression composed of noncommuting operators in terms of commuting operators. We make the following asssumptions: $p \in k{<}E_1, \ldots, E_M{>}$ is of the form

$$p = \sum_{i=1}^{l} p_i,$$

where each term p_i is of degree m; the cost of a multiplication is one unit and the cost of a differention is one unit; the cost of an addition is zero units; and the cost of adding a node to a tree is one unit, so that the cost of building a tree $t \in LT_m(E_1, \ldots, E_M)$ is m units. The proofs of the following three propositions are straightforward.

Proposition 10 *$L^{\#}$ contains $l\,m!\,N^m$ terms and, the cost of computing $L^{\#} = \chi(p)$ is $2lm\,m!\,N^m$.*

Proposition 11 *The cost of computing $\sum_{i=1}^{l} \phi(p_i)$ is $lm\,m!$.*

Proposition 12 *Let $\sigma = \phi(p)$, and denote by $|\sigma|$ the number of labeled trees with non-zero coefficients in σ. Then the cost of computing $\psi(\sigma)$ is $2m|\sigma|N^m$.*

Combining these three propositions gives

Theorem 13 *Under the assumptions above, $\mathrm{Cost}_{\mathrm{NAIVE}}(p)$, the cost of computing*

$$L^{\#} = \chi(p) = \sum_{i=1}^{l} \chi(p_i)$$

is $2lm\,m!\,N^m$. $\mathrm{Cost}_{\mathrm{BETTER}}(p)$, the cost of computing

$$L = \psi \circ \phi(p)$$

is $lm\,m! + 2m|\sigma|N^m$.

Theorem 1 now follows.

Using Theorem 7, it is possible to construct even more efficient algorithms for computing the action of elements of $k<E_1, \ldots, E_M>$ which are known to be derivations: in this case, trees for which the root has more than one child need not even be constructed.

References

[1] B. Buchberger et. al. , "Computer algebra: symbolic and algebraic computation," Springer, Wien, 1983.

[2] B. Buchberger and R. Loos, *Algebraic Simplification,* in "Computer algebra: symbolic and algebraic computation," ed. B. Buchberger et. al., Springer, Wien, 1983, 11 - 43.

[3] B. F. Caviness, *On canonical forms and simplification,* J. Assoc. Computing Machinery **17** (1970), 385-396.

[4] R. Grossman, *Evaluation of expressions involving higher order derivations,* Center For Pure and Applied Mathematics, PAM - 367, University of California, Berkeley.

[5] N. Jacobson, "Lie algebras," Interscience, New York, 1962.

[6] J. W. Milnor and J. C. Moore, *On the structure of Hopf algebras,* Ann. Math. (2) **81** (1965), 211–264.

[7] M. Sweedler, "Hopf Algebras," W. A. Benjamin, New York, 1969.

[8] R. E. Tarjan, "Data Structures and Network Algorithms," SIAM, Philadelphia, 1983.