Semi-annual Status Report for
NASA Grant NAG 2 - 513

Title: The Symbolic Computation and Automatic Analysis of Trajectories

A grant from the

National Aeronautics and Space Administration
Ames Research Center
Moffet Field, California 94035

to the

Mathematics, Statistics, and Computer Science Department
Mail Code 249, Box 4348
University of Illinois at Chicago
Chicago, Illinois 60680

Principal Investigator:	Robert Grossman
Period for Report:	July 1, 1990 - December 31, 1990
Date of Report:	April, 1991

[illegible]
Robert Grossman
Principal Investigator
(312) 996-3041

[illegible]
John Baldwin
Professor of Mathematics and
Acting Head of Department
(312) 996-3041

Karen K. Hitchcock
Karen R. Hitchcock
Vice Chancellor for Research
(312) 996-6174

(NASA-CR-188159) THE SYMBOLIC COMPUTATION AND AUTOMATIC ANALYSIS OF TRAJECTORIES Semiannual Status Report, 1 Jul. - 31 Dec. 1990 (Illinois Univ.) 55 p CSCL 12A

N91-28787
--THRU--
N91-28790
Unclas
G3/64 0002360

Semi-Annual Status Report for NASA Grant NAG2-513
The Symbolic Computation and Automated Analysis of Trajectories

Robert Grossman
University of Illinois at Chicago

March, 1991

Contents

1 Introduction

This is the semi-annual status report for NASA grant NAG2-513, "The Symbolic Computation and Automated Analysis of Trajectories." This report covers the period July 1, 1990 through December 31, 1990. The research supported by this grant is broadly concerned with the symbolic computation and mixed numeric-symbolic computation of trajectories of dynamical systems, especially control systems. The NASA Technical Officer for this grant is Dr. George Meyer, NASA Ames Research Center, Mail Stop 210-3, Moffet Field, California, 94035. In Section 2, we review the progress during the past six months. Section 3 contains bibliographic references for the articles related to this grant that were completed during this period.

2 Review of Work During Report Period

2.1 Symbolic computation of vector fields and analytic algorithms

This section is the introduction to [1], with a few minor modifications.

During the report period, we further developed the algorithms in [7] and [10] for rewriting expressions involving differential operators. The differential operators that we have in mind arise in the local analysis of nonlinear control systems. During this period, we extended these algorithms in three different directions:

1. We generalize the algorithms so that they apply to differential operators on groups. This generalization is important for applications. For example, the nonlinear system describing a robotic joint or a satellite evolves on the group $G = SO(3)$ of spatial rotations. The local study of such systems requires the computation of expressions consisting of differential operators on G.

2. We develop the data structures and algorithms to compute symbolically the action of differential operators on functions. Again, this is crucial for applications. For example, it is crucial to our approach for deriving the conditions for a numerical algorithm to remain constrained to a group. In other words, if $x_{n+1} = T(x_n)$ is the update rule for a numerical algorithm evolving on a group G, we would like to choose T so that $x_n \in G$ implies $x_{n+1} \in G$.

3. We showed that the algebraic formalism uderlying computations of this type can be given the structure of a quantum group.

For a further discussion of applications of these algorithms, see [4] and [6] and the references given there.

Here is the

Setup.

1. Let k denote either the real or complex numbers.
2. Let G denote a finite dimensional Lie group over k, g denote its Lie algebra, and $Y_1, \ldots, Y_N$ a basis for g of left-invariant vector fields.
3. Let $R = C^\infty(G)$ denote the algebra of smooth functions on G taking values in k.
4. Fix M derivations of R of the form
$$F_j = \sum_{\mu=1}^{N} a_j^\mu Y_\mu, \quad a_j^\mu \in R, \quad j = 1, \ldots, M, \tag{1}$$
and let A denote the free associative algebra $k<F_1, \ldots, F_M>$ of differential operators generated by $F_1, \ldots, F_M$, with coefficients from k.

We are concerned with the following

Problem. Given a differential operator $p \in A$ and a function $f \in R$, substitute the Equations (1) and compute $p \cdot f$ using as few operations as possible. This problem is interesting since in many cases cancellations take place.

Example 1. Let $G = \mathbf{R}^N$ denote the abelian group,
$$Y_j = \frac{\partial}{\partial x_j}, \quad j = 1, \ldots, N,$$
the (left invariant) coordinate vector fields, and F_1, F_2, F_3 three fixed vector fields defined in terms of the Y_μ via Equations (1). Then the naive substitution of (1) and simplification of $p \cdot f$, where
$$p = F_3F_2F_1 - F_3F_1F_2 - F_2F_1F_3 + F_1F_2F_3 \in A, \quad f \in R,$$
yields $24N^3$ terms, while more specialized algorithms need only compute the $6N^3$ terms which don't cancel. These types of examples are considered in [7] and [10].

Example 2. Consider the local analysis of a nonlinear system of the form

$$\dot{x}(t) = F(x(t)), \quad x(0) = x^0 \in G, \tag{2}$$

where

$$F = \sum_{j=1}^{M} u_j F_j.$$

In practice, the u_j are constants, functions of time, or perturbation parameters. The study of this system typically involves the computations of various series in the algebra A of differential operators. For example, the local flow of the system is determined by the Taylor series

$$\exp hF = 1 + hF + \frac{h^2}{2!}F^2 + \frac{h^3}{3!}F^3 + \ldots \in A[[h]].$$

An alternative to computing higher derivatives F^k is to choose constants c_i, c_{ij}, $i = 1, \ldots, k$, $j < i$, so that the expression

$$\exp hc_k\bar{F}_k \cdots \exp hc_1\bar{F}_1,$$

where

$$\begin{aligned}
\bar{F}_1 &= \sum_{\mu=1}^{N} a^\mu(x^0)Y_\mu \in g \\
\bar{F}_2 &= \sum_{\mu=1}^{N} a^\mu(\exp(hc_{21}\bar{F}_1) \cdot x^0)Y_\mu \in g \\
\bar{F}_3 &= \sum_{\mu=1}^{N} a^\mu(\exp(hc_{32}\bar{F}_2) \cdot \exp(hc_{31}\bar{F}_1) \cdot x^0)Y_\mu \in g, \\
&\vdots
\end{aligned}$$

is equal to $\exp hF$ to order k. Notice that the left invariant vector fields $\bar{F}_j$ arise by "freezing the coefficients" of F at various points along its flow.

Expanding these expressions around the common base point $x^0 \in G$ yields many terms, which must cancel in the end if the algorithm is going to approximate the flow of the underlying nonlinear system. The action of the differential operators $\bar{F}_j$ on the coefficient functions a_k^μ must also be computed. Notice, that unlike Example 1, the Y_μ here do not commute.

The computations in both examples are easily kept track of by using finite rooted trees, labeled with the symbols $F_1, \ldots, F_M$. It turns out the

the vector space, with basis the set of such trees, has an algebraic structure B which is crucial to efficiently organizing the computation. The advantage of working with the trees B is that many terms which cancel in the end need not be computed. See [6] for an expository treatment of this idea. The key observation required for this work is that it is possible to define an action of the algebra B of finite rooted trees, labeled with $F_1, \ldots, F_M$, on the ring of functions R which enjoys essentially all the properties of the familiar action of the algebra A of differential operators generated by $F_1, \ldots, F_M$ on R. It turns out that B is a Hopf algebra, just as A is, and that both actions give R the structure of what is called an H-module algebra. It also turns out that it is possible to define an alternative structure on both A and B which gives these algebras the structure of a quantum group.

2.2 Very Large Databases of Control Trajectories

During the reporting period, we completed an initial design of the system architecture for software to analyze nonlinear control systems using database computing, as described in [3] and [5]. The system will support a variety of *objects* suitable for analyzing nonlinear systems, including

1. trajectory segment
2. state
3. and control system.

The software will consist of the following components:

1. a storage manager responsible for managing the storage, retrieval and update of the objects supported by the system;
2. an object manager responsible for keeping track of the various objects recognized by the system and their relations;
3. a query manager responsible for parsing queries and generating the necessary code for the object manager so that the appropriate objects can be retrieved.

We expect software providing a "proof of concept" for this approach to be completed sometime during the coming year.

3 Articles Written During the Reporting Period

In this section we include the title pages of the articles resulting from the research supported by this grant during the reporting period.

Using Trees To Compute Approximate Solutions to Ordinary Differential Equations Exactly

October, 1990

Abstract

In this paper, we review some recent work relating families of trees to symbolic algorithms for the exact computation of series which approximate solutions of ordinary differential equations. It turns out that the vector space whose basis is the set of finite, rooted trees carries a natural multiplication related to the composition of differential operators, making the space of trees an algebra. This algebraic structure can be exploited to yield a variety of algorithms for manipulating vector fields and the series and algebras they generate.

Acknowledgments

Robert Grossman was supported in part by NASA Grant NAG 2-513 and by NSF Grant DMS-8904740.

Status

Computer Algebra and Differential Equations, M. F. Singer, editor, Academic Press, New York, 1991, in press.

Bialgebra deformations of certain universal enveloping algebras

Robert Grossman, University of Illinois at Chicago
Dave Radford, University of Illinois at Chicago

October, 1990

Abstract

Let A denote a bialgebra over a field k and let $A_t = A[[t]]$ denote the ring of formal power series with coefficients in A. Assume that A is a free algebra over k with a basis of primitives. We give a simple construction which makes A_t a bialgebra deformation of A. Usually A_t is neither commutative nor cocommutative. This construction yields deformations of bialgebras associated with families of trees.

Acknowledgments

Robert Grossman was supported in part by NASA Grant NAG 2-513 and by NSF Grant DMS-8904740.

Status

Submitted for publication.

Computations involving differential operators and their actions on functions

Peter Crouch, Arizona State University
Robert Grossman, University of Illinois at Chicago
Richard G. Larson, University of Illinois at Chicago

December, 1990

Abstract

In this paper, we further develop the algorithms described in the paper "The symbolic computation of derivations using labeled trees," R. Grossman and R. Larson, *J. of Symbolic Computation*, to appear, for rewriting expressions involving differential operators. The differential operators that we have in mind arise in the local analysis of nonlinear dynamical systems. In this work, we extend these algorithms in two different directions: We generalize the algorithms so that they apply to differential operators on groups and we develop the data structures and algorithms to compute symbolically the action of differential operators on functions. Both of these generalizations are needed for applications. This paper is preliminary: a final paper containing proofs and a further analysis of the algorithm will appear elsewhere.

Acknowledgments

Peter Crouch was supported in part by the NSF grant INT-8914643. Robert Grossman was supported in part by NASA Grant NAG 2-513 and by NSF Grant DMS-8904740. Richard Larson was supported by NSF Grant DMS-8904740.

Status

Proceedings of 1991 International Symposium on Symbolic and Algebraic Computation, ACM, 1991, in press.

References

[1] P. Crouch, R. Grossman, and R. G. Larson, "Computations involving differential operators and their actions on functions," *Proceedings of 1991 International Symposium on Symbolic and Algebraic Computation*, ACM, 1991, in press.

[2] P. Crouch, R. Grossman and R. Larson, "Trees, bialgebras and intrinsic numerical algorithms," submitted for publication.

[3] R. Grossman, "Querying databases of trajectories of differential equations I: data structures for trajectories," *Proceeding of the Twenty-Third Annual Hawaii International Conference on Systems Science*, IEEE, Washington, 1990, pp. 18–24.

[4] R. Grossman, "The evaluation of expresssions involving higher order derivations," *Journal of Mathematical Systems, Estimation, and Control*, Vol. 1 (1991), pp. 91–106.

[5] R. Grossman, "Querying databases of trajectories of differential equations II: index functions," *Proceedings of the Fourth NASA Workshop on Computational Control of Flexible Aerospace Systems*, to appear.

[6] R. Grossman, "Using trees to compute approximate solutions of ordinary differential equations exactly," *Computer Algebra and Differential Equations*, M. F. Singer, editor, Academic Press, New York, 1991, in press.

[7] R. Grossman and R. Larson, "Labeled trees and the efficient computation of derivations," in *Proceedings of 1989 International Symposium on Symbolic and Algebraic Computation*, ACM, 1989, pp. 74–80.

[8] R. Grossman and R. Larson, "Labeled trees and the algebra of differential operators," *Algorithms and Graphs*, B. Richter, editor, American Mathematical Society, Providence, 1989, pp. 81–87.

[9] R. Grossman and R. Larson, "Hopf algebraic structures of families of trees," *J. Algebra*, Vol. 26 (1989), pp. 184–210.

[10] R. Grossman and R. Larson, "The symbolic computation of derivations using labeled trees," *Journal of Symbolic Computation*, to appear.

P-25

N91-28788

Using Trees To Compute Approximate Solutions to Ordinary Differential Equations Exactly

ROBERT GROSSMAN

Laboratory for Advanced Computing
Technical Report LAC90-R24

Department of Mathematics, Statistics,
and Computer Science
University of Illinois at Chicago (M/C 249)
P. O. Box 4348
Chicago, IL 60680

October, 1990

Using Trees To Compute Approximate Solutions to Ordinary Differential Equations Exactly

Robert Grossman[*]
University of Illinois at Chicago

October, 1990

Abstract

In this paper, we review some recent work relating families of trees to symbolic algorithms for the exact computation of series which approximate solutions of ordinary differential equations. It turns out that the vector space whose basis is the set of finite, rooted trees carries a natural multiplication related to the composition of differential operators, making the space of trees an algebra. This algebraic structure can be exploited to yield a variety of algorithms for manipulating vector fields and the series and algebras they generate.

1 Introduction

In this paper, we review some recent work relating families of trees to symbolic algorithms for the exact computation of series which approximate solutions of ordinary differential equations. It turns out that the vector space whose basis is the set of finite, rooted trees carries a natural multiplication related to the composition of differential operators, making the space of trees an algebra. This algebraic structure can be exploited to yield a variety of algorithms for manipulating vector fields and the series and algebras they generate.

In Section 3, we introduce and explore the algebraic structure of trees. Section 4 describes a simplification algorithm for the rewriting of symbolic

[*]Laboratory for Advanced Computing, Department of Mathematics, Statistics, and Computer Science, Mail Code 249, University of Illinois, Chicago IL 60680, grossman@uicbert.eecs.uic.edu. This research was supported in part by the grants NASA NAG2-513 and NSF DMS-8904740.

expressions involving vector fields. Section 5 describes an algorithm for generating explicitly integrable flows associated with nilpotent Lie algebras. Section 6 exploits the relation between Taylor series and trees to study a class of intrinsic numerical integrators. We begin in Section 2 with some background.

The results surveyed here are the work of a variety of mathematicians: I especially want to mention the contributions of my collaborators Peter Crouch, Matthew Grayson and Richard Larson. The work on algebras of trees and its applications to symbolic computation is joint work with Richard Larson. All of the algorithms described here rest upon this foundation. The work on explicitly integrable flows and nilpotent Lie algebras is joint work with Matthew Grayson. The work on numerical algorithms evolving on groups is joint work with Peter Crouch.

2 Background

Consider a differential equation

$$\dot{x}(t) = E_1(x(t)) + u\, E_2(x(t)), \qquad x(0) = x^0 \in \mathbf{R}^N, \tag{1}$$

where E_1 and E_2 are vector fields and u is a parameter. In applications, u will be either a small perturbation $u = \epsilon$, a control $t \rightarrow u(t)$, or simply the constant $u \equiv 1$. Unless the vector fields E_1 and E_2 are very special, no algorithm is known which will return the general solution to the system in closed form. Our objective is to find efficient algorithms to compute various approximate solutions of the differential equation exactly using symbolic computation.

Although the impact of symbolic computation in this area is recent, the connection between the existence of closed form solutions and the approximation of general solutions is a traditional theme, dating back to at least the the nineteenth century. One can distinguish two approaches. One, championed by Lie, is based upon algebra and geometry and concerns us here; the other, championed by Weirstrass and Poincaré, is based upon complex function theory.

Consider a group of transformations acting on $\mathbf{R}^N$ of the form

$$\Phi_s : x_\mu = f_\mu(x_1, \ldots, x_N; s_1, \ldots, s_r), \qquad \mu = 1, \ldots, N,$$

with the property that the group permutes the solutions of the nonlinear system (1). Lie asked the question [51] and [52], *How can information about* *the tranformation group be used to help integrate the differential equation?* To answer this question, Lie introduced the *infinitesimal generators* of the group

$$A_k(x) = \sum_{\mu=1}^{N} \frac{\partial f_\mu}{\partial s_k} \frac{\partial}{\partial x_\mu}, \quad 1 \le k \le r$$

and showed that the A_k satisfy

$$[A_i, A_j] = \sum_{k=1}^{r} c_{ij}^k A_k,$$

where $[\cdot,\cdot]$ is the commutator, or *Lie bracket,*

$$[A_i, A_j] = A_i A_j - A_j A_i,$$

and the c_{ij}^k are constants. For example, Lie showed that if there is a one parameter group of transformations permuting the solutions of a nonlinear system in the plane (x,t), then the integrating factor for the equation may be read off from the infinitesimal generator.

Since Lie's time, this basic question has contributed to the development of a number of different fields:

- The vector fields A_j generate a filtered Lie algebra, which is usually infinite dimensional, and is the infinitesimal version of the continuous pseudogroup of transformations generated by the Φ_s. Prior work has focused on the geometry and structure theory of these algebras; important contributions have been made by Guillemin and Sternberg [37] and [36], and Hermann [52], [53], building upon the earlier work of Cartan, Ehresmann and Spencer.

- Formal sums of iterated powers of vector fields, or Lie series, have been developed by Gröbner [25], [26] and Knapp and Wanner [46], [47] into an operational calculus and used to approximate the solutions of differential equations. Lie transform methods have also been used in perturbation theory by Rand [59] and Meyer [54], in celestial mechanics by Deprit [18], and in particle physics by Dragt [20].

- Explicit series computations of solutions of differential equations have a number of interesting connections with combinatorics. Chen [13] makes uses of the shuffle product, Joyal [45], Labelle [49], Leroux [50], and Viennot [71] employ trees and species, while Rota, Krahaner and Odlyzko [62] exploit the umbral calculus.

- Ritt and Kolchin, building upon earlier work of Picard and Vessiot, developed the field of differential Galois theory. The goal is to obtain a theory describing the solvability of differential equations analogous to Galois' theory describing the solvability of algebraic equations. The survey by Singer [67] provides a good description of this field from the viewpoint of symbolic computation. Other relevant contributions include the beginnings of a differerential Gröbner theory [3], [38], analogous to the Gröbner theory in commutative algebra; and a Hopf algebraic interpretation of Picard-Vessiot theory by Takeuchi [70].
- There is now a resurgence of interest in using symmetry groups to help integrate differential equations. This direction of research has been active in the Soviet Union for some time, but during the past decade there has been increased interest in the United States. Important contributions have been made by Olver [57], Schwarz [64], and Bluman and Cole [5]. Closely related is the study by Caviness [66] of conservation laws for differential equations.

During the past several years, there has been increasing interest in symbolic computation and differential equations. Work has proceeded in a number of directions, and is based upon both the Lie and the Weirstrass and Poincaré traditions:

- Zippel [73] is writing a modular symbolic computation system which supports the ability to call high quality numerical routines. Using such a system, he has shown how symbolic algorithms can be used to select appropriate numerical algorithms.
- Guckenhemier [35] has produced the program kaos which allows the user to access a variety of algorithms to investigate a differential equation from the point of view of modern dynamical systems.
- There are a number of programs to compute symmetry groups of differential equations, including one written by Char [12], and the programs SODE and SPDE written by Schwarz [64].
- Abelson and Sussman and their group at MIT [1], [2] have combined techniques in artificial intelligence to produce software which automatically analyzes the qualitative features of a differential equation.
- Wang [72] and Steinberg [68] have developed systems which use symbolic computation to produce high quality, optimized Fortran code to solve differential equations.

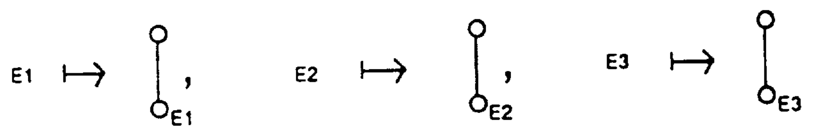

Figure 1: The trees associated with the vector fields E_j.

- Della-Dora and Tournier [17] have used the fundamental ideas of Ramis [58] to produce a system to analyze linear ordinary differential equations. They are now turning their attention to nonlinear systems.

Point of view. The point of view taken here is to focus on the algorithmic aspects of the computation of the vector fields A_j and their brackets and to use this information to develop appropriate algorithms which use exact symbolic techniques to approximately integrate the trajectories of the differential equation. As will become clear, there are a number of interesting points of contact between this approach and the approaches just described.

In the following sections, we review data structures and algorithms for the symbolic computation of the flows of vector fields, and for the symbolic approximation of general flows by flows which can be studied symbolically.

3 Vector fields and the algebra of Cayley trees

In this section, we describe a data structure which is central to the algorithms we give for the symbolic computation of series which approximate the solutions of differential equations. The basic idea is to assign trees to vector fields as illustrated in Figure 1, and then to impose a multiplication on trees which is compatible with the composition of vector fields.

Consider three vector fields

$$E_1 = a_1 D_1 + \cdots a_N D_N, \qquad E_2 = b_1 D_1 + \cdots b_N D_N,$$

$$E_3 = c_1 D_1 + \ldots c_N D_N$$

where $D_i = \partial / \partial x_i$, and a_i, b_i and c_i are smooth functions on $\mathbf{R}^N$. Now

$$E_2 \cdot E_1 = \sum b_j (D_j a_i) D_i + \sum b_j a_i D_j D_i$$

and $E_3 \cdot E_2 \cdot E_1$ is equal to

$$\sum c_k (D_k b_j)(D_j a_i) D_i + \sum c_k b_j (D_k D_j a_i) D_i + \sum c_k b_j (D_j a_i) D_k D_i$$

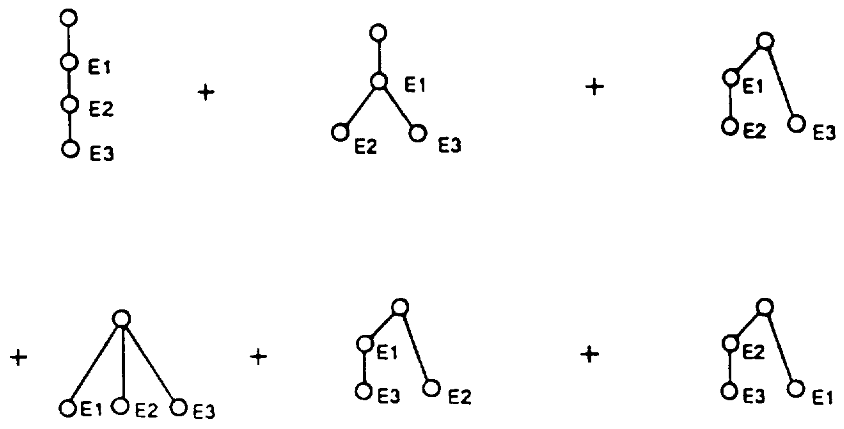

Figure 2: The trees associated with Equation 2.

$$+\sum c_k b_j a_i D_k D_j D_i + \sum c_k b_j (D_k a_i) D_j D_i + \sum c_k (D_k b_j) a_i D_j D_i. \quad (2)$$

Here the sum is for $i, j, k = 1, \ldots, N$ and hence involves $O(N^3)$ differentiations. It is convenient to keep track of the terms that arise in this way using labeled trees: we indicate in Figure 2 the trees that are associated with the six sums in this expression.

An expression such as

$$[E_3, [E_2, E_1]] = E_3E_2E_1 - E_3E_1E_2 - E_2E_1E_3 + E_1E_2E_3 \quad (3)$$

gives rise in this fashion to 24 trees corresponding to the $24N^3$ differentiations that a naive computation of this expression requires. On the other hand, 18 of the trees cancel, saving us from computing $18N^3$ terms. We are left with $6N^3$ terms of the form $(junk)D_{\mu_1}$. A careful examination of this correspondence between labeled trees and expressions involving the E_j's shows that the composition of the vector fields E_j's, viewed as first order differential operators, corresponds to a multiplication on trees. This multiplication is illustrated in Figure 3. It turns out that this construction yields an algebra, which we call the ***algebra of Cayley trees***.

Here is a more precise description for the specialist, following [27] and [30]. Let k denote a field of characteristic 0. We say that a finite rooted tree is labeled with $\{E_1, \ldots, E_M\}$ in case each node, except for the root, is assigned a element from this set. Let $\mathcal{LT}(E_1, \ldots, E_M)$ denote the set of labeled trees and let $k\{\mathcal{LT}(E_1, \ldots, E_M)\}$ denote the vector space whose basis consists of labeled trees in $\mathcal{LT}(E_1, \ldots, E_M)$. Suppose that t_1,

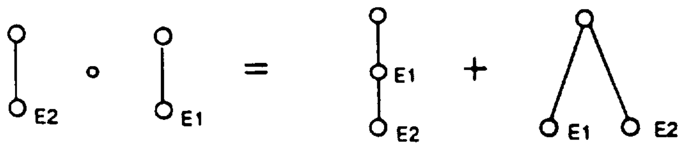

Figure 3: An example of multiplying two trees.

$t_2 \in \mathcal{LT}(E_1, \ldots, E_M)$ are 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 tree t_2 by making each s_i the child of some node of t_2. The product t_1t_2 is defined to be the sum of these $(n+1)^r$ trees. It can be shown that this product is associative, and that the trivial tree consisting only of the root is a right and left unit for this product. In [27], we define a comultiplication on $k\{\mathcal{LT}(E_1, \ldots, E_M)\}$ and show that

Theorem 3.1 *The vector space $k\{\mathcal{LT}(E_1, \ldots, E_M)\}$ with basis all equivalence classes of finite rooted trees is a cocommutative graded connected Hopf algebra.*

We call this algebra the *algebra of Cayley trees* generated by the set of labeled trees $\mathcal{LT}(E_1, \ldots, E_M)$. The relation between trees and differential operators goes back to Cayley [8], [9]. The coalgebra structure on this space is very similar to the coalgebra structure defined by Joni and Rota [44]. However, the coalgebra structure defined there was defined for individual combinatorial objects, rather than for a class of objects such as the family of rooted trees. Butcher [6] and [7] has also defined a multiplication on the vector space which is dual to the space of trees. This multiplication is closely related to the one just defined.

4 Symbolic evaluation of vector field expressions

When expressions involving vector fields, such as Lie brackets, are written out in coordinates, there is typically a lot of cancellation. Similar cancellation occurs in expressions involving Poisson brackets and when flows are concatenated, as in Campbell-Baker-Hausdorf expansions. In this section, we use the algebra of Cayley trees to exploit this cancellation in order to compute more efficiently formal expressions involving vector fields.

This is the set up. Let R denote a ring of functions. In applications, R is usually either the ring of polynomial functions, rational functions, or C^∞

functions. Fix several first order differential operators with coefficients from R

$$E_j = \sum_{\mu=1}^{N} a_j^\mu D_\mu, \quad a_j^\mu \in R, \quad j = 1, \dots, M \tag{4}$$

that are defined in terms of a basis of first order differential operators

$$D_\mu = \frac{\partial}{\partial x_\mu}, \quad \mu = 1, \dots, N.$$

Simplifying any expression in the E_j's using Equation (4) yields a differential operator, which we can view as an element of End R.

We now define a homomorphism from the algebra of expressions in the E_j's to the algebra of trees. Indeed, the assignment in Figure 1 extends to an algebra homomorphism from the free associate algebra in the symbols E_j to the algebra of Cayley trees $k\{\mathcal{LT}(E_1, \dots, E_M)\}$. It is straightforward to define a downward-pointing arrow so that the following diagram commutes:

$$\begin{tikzcd} k\langle E_1, \dots, E_M\rangle \arrow[r] \arrow[dr] & k\{\mathcal{LT}(E_1, \dots, E_M)\} \arrow[d] \\ & \text{End } R \end{tikzcd} \tag{5}$$

Algorithm 4.1 *To rewrite expressions in the first order differential operators E_j in terms of the basis*

$$\frac{\partial}{\partial x_{\mu_1}}, \quad \frac{\partial^2}{\partial x_{\mu_1} \partial x_{\mu_2}}, \dots, \quad \mu_1, \mu_2, \dots = 1, \dots, N,$$

compute the composition of the rightward and downward pointing arrows in the diagram above.

In [28], [29] and [33], we show that the algorithm is much more efficient than naive substitution, which corresponds to computing the diagonal arrow directly. In some common cases, the improvement in efficiency is exponential. We have implemented this and related algorithms in Maple, Mathematica and Snobol.

We illustrate this algorithm by working the example of the last section following [31]: consider a higher order derivation of the form

$$p = E_3 E_2 E_1 - E_3 E_1 E_2 - E_2 E_1 E_3 + E_1 E_2 E_3.$$

Naive simplification requires computing $24N^3$ terms of the form described in Table 1. The image of p in the algebra of Cayley trees contains 24

No. of terms	Form of terms
$8N^3$	coeff. D_{μ_1}
$12N^3$	coeff. $D_{\mu_2} D_{\mu_1}$
$4N^3$	coeff. $D_{\mu_3} D_{\mu_2} D_{\mu_1}$

Table 1: Naive computation of the differential operator corresponding to p.

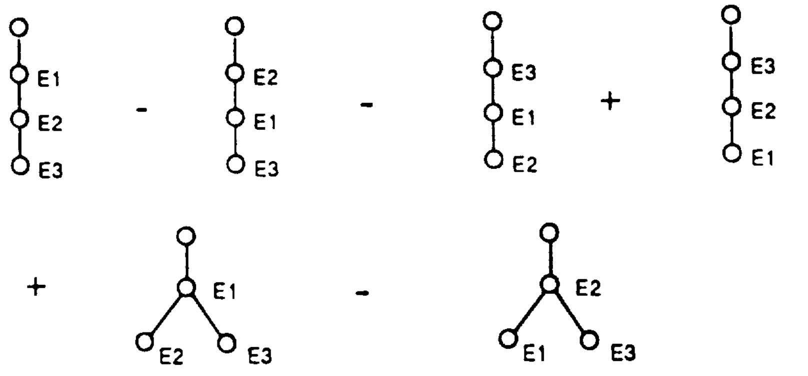

Figure 4: The surviving labeled trees.

trees, six for each of the four terms of p. For example, the six labeled trees corresponding to the first term are given in Figure 2. Eighteen of these trees cancel, leaving the six trees in Figure 4. The corresponding differential operator is equal to

$$\begin{aligned}
&\sum a_3^{\mu_3}(D_{\mu_3} a_2^{\mu_2})(D_{\mu_2} a_1^{\mu_1}) D_{\mu_1} - \sum a_3^{\mu_3}(D_{\mu_3} a_1^{\mu_2})(D_{\mu_2} a_2^{\mu_1}) D_{\mu_1} \\
&- \sum a_2^{\mu_3}(D_{\mu_3} a_1^{\mu_2})(D_{\mu_2} a_3^{\mu_1}) D_{\mu_1} + \sum a_1^{\mu_3}(D_{\mu_3} a_2^{\mu_2})(D_{\mu_2} a_3^{\mu_1}) D_{\mu_1} \\
&+ \sum a_3^{\mu_3} a_2^{\mu_2}(D_{\mu_3} D_{\mu_2} a_1^{\mu_1}) D_{\mu_1} - \sum a_3^{\mu_3} a_1^{\mu_2}(D_{\mu_3} D_{\mu_2} a_2^{\mu_1}) D_{\mu_1},
\end{aligned}$$

and contains $18N^3$ fewer terms of the form indicated in Table 2 than does the naive computation of p. An example of the cancellation of labeled trees is given in Figure 5.

To summarize: we have defined an algebraic structure on families of trees which mirrors the algebraic structure of formal expressions in the variables E_j, but which alleviates the need for computing intermediate expressions which cancel when the noncommuting E_j's are expressed in terms of the

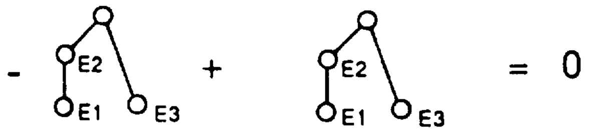

Figure 5: The term $E_3E_1E_2$ contributes the first labeled tree and the term $E_1E_2E_3$ contributes the second, which cancel.

No. of terms	Form of terms
$2N^3$	coeff. D_{μ_1}
$12N^3$	coeff. $D_{\mu_2}D_{\mu_1}$
$4N^3$	coeff. $D_{\mu_3}D_{\mu_2}D_{\mu_1}$

Table 2: Terms in the computation p which cancel.

commuting D_μ's. In the following sections, we will see other expressions of this simple idea.

Algorithm 4.1 can be extended in several different directions.

1. The examples above concern vector fields defined on $\mathbf{R}^N$. It is possible to work out similar algorithms for vector fields defined on more general objects, such as Lie groups. This is important for applications in robotics and rigid body dynamics. For example, the group $G = SO(3)$ is the appropriate configuration space for a rotating rigid body. To be more specific, assume the vector fields are of the form

$$E_j = \sum_{\mu=1}^{N} a_j^\mu Y_\mu,$$

   where the Y_μ are left-invariant vector fields on G and the a_j^μ are smooth functions on the group. In this case, the Cayley algebras are generated by *ordered*, labeled trees [16]. Roughly speaking, the trees are ordered since the vector fields Y_μ no longer commute.

2. The natural action of differential operators on functions turns the ring of functions R into a module. In this same way, the trees have a natural action on R, as indicated in the Diagram 5. It turns out [34] that this gives R the structure of K/k-bialgebra, as introduced by Nichols [55]

and [56]. These types of algebras are closely related to differential algebras.

3. It is a basic fact that the local properties of the nonlinear system (1) are determined by the algebraic properties of the higher order iterated Lie brackets; see, for example, [40]. Unfortunately, due to intermediate expansion swell, it is often difficult to compute these using current computer algebra systems. It turns out that higher order Lie brackets not only involve the cancellation of all terms above the first order but also the cancellation of some of the first order terms. Algorithm 4.1 can be used so that the terms arising in these first order cancellations need not be computed.

5 Exponentials, Lie brackets, and nil flows

Some differential equations have the property that their flows can be integrated symbolically in closed form. For example, this holds for differential equations of the form (1), if E_1 and E_2 are homogeneous in the appropriate sense and generate a graded, nilpotent Lie algebra. In this section, we give an algorithm which, given an appropriate nilpotent Lie algebra, returns vector fields on $\mathbf{R}^N$ with polynomial coefficients which generate the Lie algebra. This leads to an interesting class of differential equations whose flows can be integrated symbolically in closed form. At the end of the section, we look at several applications of this algorithm.

By the third fundamental theorem of Lie [63], we know that a nilpotent Lie algebra arises from some Lie algebra of vector fields. What is not obvious is how to construct such vector fields. Nilpotent Lie algebras of vector fields have been used as an important tool in partial differential equations by Folland and Stein [21], Rothschild and Stein [60], and Rockland [61]; and in control theory by Krener [48], Hermes [41], [42], and Crouch [15], [14]. We will see below how they are also a useful tool in developing symbolic-numeric algorithms to integrate flows.

Our goal is to describe a natural representation of nilpotent Lie algebras on vector fields on Euclidean space with polynomial coefficients. To define this represenation, we define a *basis of Hall trees* on generators E_1 and E_2 recursively as follows:

1. basis elements consist of rooted, binary trees, with all nodes, except the root, labeled with E_1, E_2, E_3, ... satisfying

   (a) all right children are leaves and labeled with either E_1 or E_2
   (b) the sequence formed by the labels of the right leaves at increasing distance from the root is nonincreasing

2. the two rooted trees consisting of a root and a single (left) child labeled E_i, for $i = 1, 2$ are in the basis and of length 1

3. if we have defined basis elements t_i of length $1, \ldots, r-1$, they are simply ordered so that $t_i < t_j$ in case the length of t_i is less than the length of t_j.

4. a tree consisting of a root with a single (left) child labeled E_i is in the basis provided that E_i's left child is in the basis and of lower order.

To each such tree corresponds an element E_i in the Hall basis [39] of the free, nilpotent Lie algebra on two generators. Figure 6 illustrates how the basis of Hall trees leads naturally to a representation of the free nilpotent Lie algebra generated by E_1 and E_2 on the space of vector fields on $\mathbf{R}^N$ with polynomial coefficients. See [22] for further details of the following algorithm:

Algorithm 5.1 *Fix $r > 1$ and say that the free, nilpotent Lie algebra of two generators of rank r has dimension N. Let $E_1, \ldots, E_N$ denote the Hall basis. Then the vector fields on $\mathbf{R}^N$ defined as in Figure 6 satisfy*

1. *the Lie algebra they generate is isomorphic to the free, nilpotent Lie algebra on two generators of rank r*

2. *any trajectory of the nonlinear system*

$$\dot{x}(t) = E_1(x(t)) + u(t)\, E_2(x(t)), \qquad x(0) = x^0 \in \mathbf{R}^N,$$

   can be computed in closed form in terms of quadratures of the function $t \to u(t)$

3. *there exist constants $\alpha_1, \ldots, \alpha_N$ such that*

$$\exp(E_2)\exp(E_1) = \exp\left(\sum_{i=1}^{N} \alpha_i E_i\right),$$

   and the α_i can be computed by solving a lower triangular linear system.

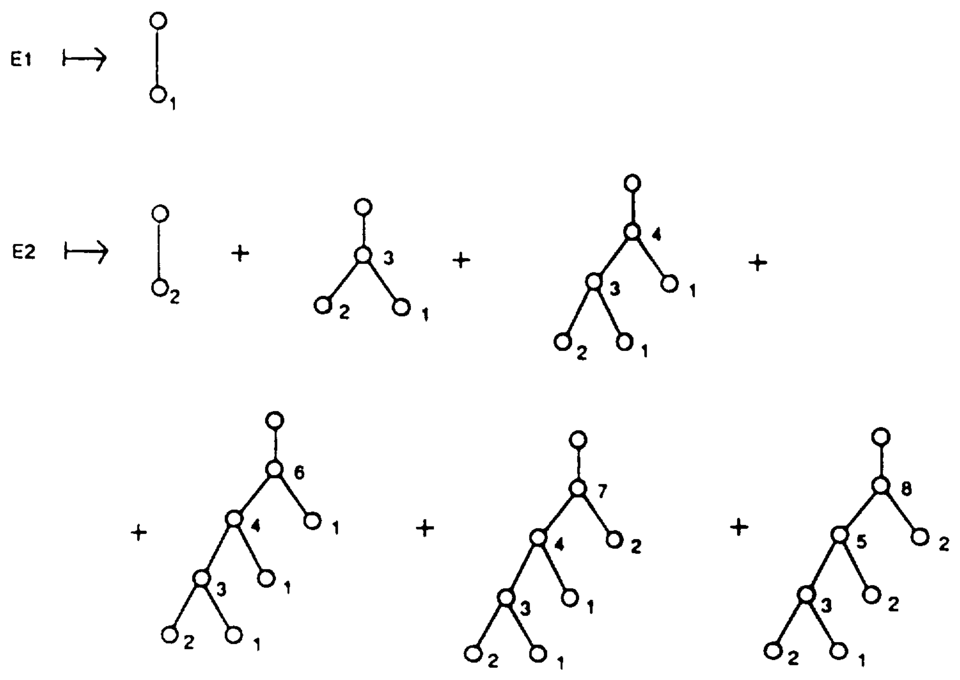

Figure 6: The vector fields arising from the basis of Hall trees. The first tree is sent to D_1, while the sum of the trees is sent to $D_2 - x_1 D_3 + \frac{1}{2} x_1^2 D_4 + x_1 x_2 D_5 - \frac{1}{6} x_1^3 D_6 - \frac{1}{2} x_1^2 x_2 D_7 - \frac{1}{2} x_1 x_2^2 D_8$.

As a note, this algorithm was found only after several months of experimentation with Maple and required the careful study of the free, nilpotent Lie algebra on two generators of rank 5, which is 23 dimensional.

We conclude this section with some remarks describing several applications of this algorithm and related work.

1. Locally approximating a nonlinear system by an explicitly integrable nilpotent one yields a number of integration algorithms, which we refer to as *piecewise nilpotent integration algorithms* (PWNI). The basic idea [23], [24] is to approximate locally a nonlinear system at a given point by an explicitly integrable nilpotent one in which the computations can be done symbolically in closed form, and to patch together the various nilpotent approximations at nearby points using a standard numerical algorithm. Preliminary work indicates that this leads to symbolic-numeric algorithms for the path planning problem and efficient algorithms to integrate neighborhoods of trajectories around a given fixed reference trajectory.

2. Algorithm 5.1 also provides an efficient means for computing the concatenation of flows. Write $x(t) = \exp E \cdot x^0$ for the flow of the nonlinear system
$$\dot{x}(t) = E(x(t)), \quad x(0) = x^0, \tag{6}$$
The Campbell-Baker-Hausdorff formula expresses the concatenation of two flows as a single flow:
$$\begin{aligned}\exp(tE_2)\exp(tE_1) = \exp(tE_2 + tE_1 &+ 1/2t^2[E_2, E_1] \\ &+ 1/12[[E_2, E_1], E_1] - 1/12[[E_2, E_1], E_2] + \cdots).\end{aligned} \tag{7}$$
Consider the equation
$$\exp(E_2)\exp(E_1) = \sum_{i=1}^{N} c_i E_i,$$
where E_i are the vector fields produced by the algorithm. Since all flows of the vector fields E_1 and E_2 are explicitly integrable in closed form, this reduces the computation of the c_i to the solution of a linear system, which is lower triangular.

3. We can also use Algorithm 5.1 to derive a class of numerical integrators, which are sometimes known as splitting methods. For example,

suppose that E_1 and E_2 are separately integrable in closed form, but $E_1 + E_2$ is not. Then using the algorithm, we can compute constants τ_i such that

$$\exp(\tau_7 E_1) \cdot \exp(\tau_6 E_2) \cdot \exp(\tau_5 E_1) \cdot \exp(\tau_4 E_2)$$
$$\cdot \exp(\tau_3 E_1) \cdot \exp(\tau_2 E_2) \cdot \exp(\tau_1 E_1) \cdot = \exp(E_1 + E_2) + O(t^5). \quad (8)$$

This formula yields a numerical integrator for our original nonlinear system (1) (with $u \equiv 1$). This algorithm is used in accelerator physics [65].

4. Recently, Strichartz [69] has shown that the solution of the initial value problem
$$\dot{x}(t) = E(t, x(t)), \quad x(0) = x^0$$
can be written as
$$x(t) = \exp(G(t))x^0,$$
with
$$G(t) \sim \sum_{r=1}^{\infty} \sum_{\sigma} c_{r,\sigma}$$
$$\cdot \int [\cdots [E(s_{\sigma(1)}, E(s_{\sigma(2)}] \cdots] E(s_{\sigma(r)}] \, ds,$$
and where σ ranges over the symmetric group on r symbols, the integration region is a simplex in $\mathbf{R}^r$, $E(s)$ denotes the vector field $E(s, \cdot)$, and exp is defined as in Equation 6. Formulas of this type date back to Chen [13] and appear to be related to Algorithm 5.1.

5. F. Bergeron, N. Bergeron, and A. M. Garsia have also exploited a relation between trees and polynomials in their study of free Lie algebras; see [4] and the references cited there.

6 Taylor series and intrinsic integrators

Although nonlinear systems often conserve quantities such as energy or angular momentum, most numerical integrators do not. Similarly, nonlinear systems typically evolve on some underlying geometric space, such as a Lie group or homogeneous space, but most numerical integrators do not remain in such a space.

Recently, there has been a flurry of activity related to numerical integrators preserving the symplectic structure, sparked off by the work of

Channell [10] and Scovel [11], and based upon earlier work of deVogeleaere [19]. These types of numerical schemes have found important applications in the long term study of orbits in accelerator physics and in other areas [65]. The derivation of these integrators typically involves the symbolic computation of Taylor series and generating series for the symplectic transformation which is the update for the numerical integrator.

Consider a numerical integrator for a differential equation

$$\dot{x}(t) = E(x(t)), \qquad x(0) = p \in M$$

evolving on a space M. Call a numerical integrator *intrinsic* in case $x_n \in M$ implies $x_{n+1} \in M$, for $n \geq 1$, where x_n is the approximation to the trajectory $x(t)$ at time t_n. One means of deriving intrinsic numerical integrators is to mimic the derivation of standard numerical integrators, but to impose additional constraints on the scheme to satisfy the added condition that the points x_n remain in the space M. This typically involves the careful study of the Taylor series of the solution.

This can be done by using the Cayley algebra of trees, as briefly indicated in [32]. As an illustration of this, we consider intrinsic Runge-Kutta type algorithms evolving on a Lie group G, following [16]. Let g denote its Lie algebra, and let $Y_1, \ldots, Y_N$ denote a basis of g . We give an algorithm to approximate solutions to differential equations evolving on G of the form:

$$\dot{x}(t) = E(x(t)), \qquad x(0) = p \in G,$$

where

$$E = \sum_{\mu=1}^{N} a^{\mu} Y_{\mu},$$

and the $=$ a^{μ} are analytic functions on G. Let $\exp(tE) \cdot x$ denote the solution $x(t)$ at time t. The algorithms depends upon constants c_i and c_{ij}, for $i = 1, \ldots, k$ and $j < i$. For fixed constants, define the following elements of the Lie algebra g

$$\begin{aligned}
\bar{E}_1 &= \sum_{\mu=1}^{N} a^{\mu}(x_n) Y_{\mu}, \in g \\
\bar{E}_2 &= \sum_{\mu=1}^{N} a^{\mu}(\exp(hc_{21}\bar{E}_1) \cdot x_n) Y_{\mu} \in g \\
\bar{E}_3 &= \sum_{\mu=1}^{N} a^{\mu}(\exp(hc_{32}\bar{E}_2) \cdot \exp(hc_{31}\bar{E}_1) \cdot x_n) Y_{\mu} \in g \ldots
\end{aligned}$$

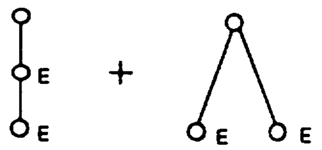

Figure 7: Trees associated with third order terms in a Taylor series.

These arise by "freezing the coefficients" of E at various points along the flow of E.

Algorithm 6.1 *Given an initial point x_0 on the group, define*

$$x_{n+1} = \exp hc_k \bar{E}_k \cdots \exp hc_1 \bar{E}_1 x_n,$$

for $n \geq 0$.

Notice that if we assume the exponential $\exp(h\bar{E}_i)$ maps the Lie algebra to the Lie group exactly, then this algorithm is intrinsic. For a group such as $G = SO(3)$, there are classical closed form expressions for the exponential map and Algorithm 6.1 yields an intrinsic integrator. Notice also that if G is the abelian group $\mathbf{R}^N$, then the algorithm becomes the classical Runge-Kutta algorithm.

The first step is to derive the equations that the coefficients c_i and c_{ij} must satisfy in order for the algorithm to yield an rth order numerical integrator. This can easily be done using the Cayley algebra of ordered trees [16]. The trees are ordered since the vector fields Y_μ do not commute.

Assume for the moment that $G = \mathbf{R}^N$ and consider the terms in the Taylor series

$$\begin{aligned} x(t+h) - x(t) &= h\dot{x}(t) + \frac{h^2}{2!}\ddot{x}(t) + \frac{h^3}{3!}x^{(iii)}(t) + \cdots \\ &= hE + \frac{h^2}{2!}DE \cdot E \\ &\quad + \frac{h^3}{3!}\left(DE \cdot DE \cdot E + D^2E(E,E)\right) + \cdots . \end{aligned}$$

Notice that there is a natural correspondence between trees and terms in the series. For example, the $h^3/3!$ terms are associated with the two labeled trees in Figure 7. This observation goes back to at least Cayley [8], [9].

We now generalize this to a Lie group following [16]. Recall that Diagram 5 induces an action of trees on the ring of analytic functions on G. Using this action, we can now state the

Lemma 6.1 *Let α denote the tree consisting of a root with a single child labeled F. Then for any analytic function f on the group and for sufficiently small t,*

$$f(\exp(tF)\cdot x) = \exp(t\alpha)\cdot f\,|_x\,.$$

Notice that if G is Euclidean space, and if the functions f are the coordinate functions $x_1, \ldots, x_n$, then this becomes the familar Taylor series.

Using this lemma, it is now easy to compute the equations that the coefficients c_i and c_{ij} must satisfy. In spirit, this is similar to Butcher's use of trees to analyze higher order Runge-Kutta algorithms in Euclidean space [6], [7].

References

[1] H. Abelson and G. J. Sussman, Dynamicists' Workbench I: "Automatic Preparation of Numerical Experiments," R. Grossman, editor, *Symbolic Computation: Applications to Scientific Computing,* SIAM, 1989.

[2] H. Abelson, M. Eisenberg, M. Halfant, J. Katzenelson, E. Sacks, G. Sussman, J. Wisdom, and K. Yip, "Intelligence in Scientific Computing," *Communications ACM*, Vol. 32, pp. 546–562, 1989.

[3] J. Apel and W. Lassner, "An extenstion of Buchberger's algorithms and calculations in enveloping algebras of Lie algebras," *J. Symbolic Computation*, Vol. 6, pp. 361–370, 1988.

[4] F. Bergeron, N. Bergeron, and A. M. Garsia, "Idempotents for the free Lie algebra and q-enumeration," *Invariant Theory and Tableaux*, D. Stanton, editor, Springer-Verlag, New York, pp.166–190, 1990.

[5] G. W. Bluman and J. D. Cole, *Similarity Methods for Differential Equations*, Springer-Verlag, New York, 1974.

[6] J. C. Butcher, "An order bound for Runge-Kutta methods," *SIAM J. Numerical Analysis*, Vol, 12, pp. 304–315, 1975.

[7] J. C. Butcher, *The Numerical Analysis of Ordinary Differential Equations*, John Wiley, 1986.

[8] A. Cayley, "On the theory of analytical forms called trees", *Collected Mathematical Papers of Arthur Cayley,* Cambridge University Press, Vol. 3, pp. 242–6, 1890.

[9] A. Cayley, "On the analytical forms called trees, second part", in *Collected Mathematical Papers of Arthur Cayley*, Cambridge University Press, Vol. 4, pp. 112–5, 1891.

[10] P. Channell, "Sympletic Integration for Particles in Electric and Magnetic Fields," *Accelerator Theory Note* No. AT-6:ATN-86-5, Los Alamos National Laboratory, 1986.

[11] P. Channell and C. Scovel, "Sympletic Integration of Hamiltonian Systems," submitted for publication.

[12] B. Char, "Using Lie transformation groups to find closed form solutions to first order ordinary differential equations," *SYMSAC '81: Proceedings of the 1981 ACM Symposium on Symbolic and Algebraic Computation*, P. Wang, editor, ACM Press, pp. 44–50, 1981.

[13] K. T. Chen, *Integration of paths, geometric invariants and a generalized Baker-Hausdorff formula*, *Annals of Mathematics*, Vol. 65, pp. 163–178, 1957.

[14] P. E. Crouch, "Dynamical Realizations of Finite Volterra Series," *SIAM Journal of Control and Optimization*, Vol. 19, pp. 177-202, 1981.

[15] P. E. Crouch, "Graded and Nilpotent Approximations to Input-Output Systems, *Nonlinear Controllability and Optimal Control*, H. J. Sussmann, editor, Marcel Dekker, New York, pp. 383–430, 1990.

[16] P. Crouch, R. Grossman, and R. Larson, "Trees, bialgebras, and intrinsic numerical integrators," *Laboratory for Advanced Computing Technical Report*, Number LAC90-R23, University of Illinois at Chicago, May, 1990.

[17] J. Della-Dora and E. Tournier, "Formal solutions of linear difference equations: method of Pincherle-Ramis," *Proceedings of the 1986 Symposium on Symbolic and Algebraic Computation*, ACM Press, New York, 1986.

[18] A. Deprit, "Canonical Transformations Depending on a Small Parameter," *Celestial Mechanics*, Vol. 1, pp 12–30, 1969.

[19] R. deVogelaere, "Methods of integration which preserve the contact transformation property of the Hamiltonian equations," *Department of Mathematics, University of Notre Dame Report* Vol. 4.

[20] A. J. Dragt and J. M. Finn, "Lie Series and Invariant Functions for Analytic Symplectic Maps," *J. Math. Physics*, Vol. 17, pp. 2215–2227, 1976.

[21] G. B. Folland and E. M. Stein, "Estimates for the $\bar{\partial}_b$ complex and analysis of the Heisenberg group," *Communications in Pure and Applied Mathematics*, Vol. 27, pp. 429–522, 1974.

[22] M. Grayson and R. Grossman, "Models for free, nilpotent lie algebras," *J. Algebra*, Vol. 35, pp. 177–191, 1990.

[23] M. Grayson and R. Grossman, "Nilpotent Lie algebras and vector fields," *Symbolic Computation: Applications to Scientific Computing*, R. Grossman, editor, SIAM, Philadelphia, pp. 77–96, 1989.

[24] M. Grayson and R. Grossman, "The simultaneous integration of trajectories using nilpotent normal forms," *Laboratory for Advanced Computing Technical Report*, Number LAC90-R19, University of Illinois at Chicago, May, 1990.

[25] W. Gröbner, *Die Lie-Reihen Und Ihre Anwendungen*, Veb Deutscher Verlag der Wissenschaften, Berlin, 1967.

[26] W. Gröbner and H. Knapp, *Contributions to the Method of Lie Series*, Hochschulskripten Bibliographisches Institut, Mannhiem, 1967.

[27] R. Grossman and R. Larson, "Hopf algebraic structures of families of trees," *J. Algebra*, Vol. 26, pp. 184–210, 1989.

[28] R. Grossman and R. Larson, "Labeled trees and the algebra of differential operators," *Algorithms and Graphs*, B. Richter, editor, American Mathematical Society, Providence, pp. 81–87, 1989.

[29] R. Grossman and R. Larson, "Labeled trees and the efficient computation of derivations," in *Proceedings of 1989 International Symposium on Symbolic and Algebraic Computation*, ACM, pp. 74–80, 1989.

[30] R. Grossman and R. Larson, "Solving nonlinear equations from higher order derivations in linear stages," *Advances in Mathematics*, Vol. 82, pp. 180–202, 1990.

[31] R. Grossman, "The evaluation of expresssions involving higher order derivations," *Journal of Mathematical Systems, Estimation, and Control*, Vol. 1, 1990.

[32] R. Grossman and R. Larson, "Hopf-algebraic structure of combinatorial objects and differential operators," *Israeli J. Math.*, to appear.

[33] R. Grossman and R. Larson, "The symbolic computation of derivations using labeled trees," *Laboratory for Advanced Computing Technical Report* Number LAC90-R09, University of Illinois at Chicago, January, 1990.

[34] R. Grossman and R. Larson, "Bialgebras of trees and differential operators," *Laboratory for Advanced Computing Technical Report* Number LAC90-R22, University of Illinois at Chicago, August, 1990.

[35] J. Guckenheimer and S. Kim, *Kaos*, to appear.

[36] V. W. Guillemin and S. Sternberg, "Infinite dimensional primitive Lie algebras," *Journal of Differential Geometry*, Vol. 4, pp. 257–282, 1970.

[37] V. W. Guillemin, "An algebraic model of transitive differential geometry," *Bulletin of the American Mathematical Society*, Vol. 70, pp. 16–47, 1964.

[38] C. Guoting, "Gröbner bases in rings of differential operators," *Research Reprints of the Institute of Systems Science, Academica Sinica*, Beijing, 1989.

[39] M. Hall, "A basis for free Lie rings and higher commutators in free groups," *Proc. Amer. Math. Soc.*, Vol. 1, pp. 575–581, 1950.

[40] R. Hermann and A. J. Krener, "Nonlinear controllability and observability," *IEEE Trans. Automatic Control*, Vol. AC-22, pp. 728–740.

[41] H. Hermes, "Control systems which generate decomposible Lie algebras," *J. Diff. Eqns.*, Vol. 44, pp. 166–187, 1982.

[42] H. Hermes, "Nilpotent approximations of control systems and distributions," *SIAM J. Control Optim.*, Vol. 24, pp. 731–736, 1986.

[43] H. Hermes, A. Lundell, and D. Sullivan, "Nilpotent bases for distributions and control systems," *J. Diff. Equations*, Vol. 55, pp. 385–400, 1984.

[44] S. A. Joni and G.-C. Rota, "Coalgebras and bialgebras in combinatorics," *Stud. Appl. Math.*, Vol. 61, pp. 93–139, 1979.

[45] A. Joyal, "Une theorie combinatorire des series formelles," *Advances in Mathematics*, Vol. 42, pp. 1–82, 1981.

[46] H. Knapp and G. Wanner "Numerical solution of ordinary differential equations by Grbner Lie series method," *Mathemtical Research Center Report* No. 880, University of Wisconson, Madison, 1968.

[47] H. Knapp and G. Wanner "LIESE, a program for ordinary differential equations using Lie series," *Mathemtical Research Center Report* No. 881, University of Wisconson, Madison, 1968.

[48] A. Krener, "Bilinear and nonlinear realizations of input-output maps," *SIAM J. Control Optim*, Vol. 13, pp. 827–834, 1975.

[49] G. Labelle, "Une novelle demonstration combinatorie des formules d'inversion de Lagrange," *Advances in Mathematics*, Vol. 42, pp. 217–247, 1981.

[50] P. Leroux and X. G. Viennot, "Combinatorial Resolution of Systems of Differential Equations, I. Ordinary Differential Equations," *Lect. Notes in Maths.*, Vol. 1234, Springer-Verlag, New York, pp. 210–245, 1986.

[51] S. Lie, "Zur theorie des integrabilitetsfaktors," *Christiania Forh.*, pp. 242–254, 1874. Reprinted *Gesamm. Abh.*, Vol. III, Number XIII, pp. 176–187.

[52] S. Lie, *Sophus Lie's 1880 Transformation Group Paper*, translated by Michael Ackerman, comments by Robert Hermann, Math Sci Press, Brookline, Massachusetts, 1975.

[53] S. Lie, *Sophus Lie's 1884 Differential Invariant Paper*, translated by Michael Ackerman, comments by Robert Hermann, Math Sci Press, Brookline, Massachusetts, 1976.

[54] K. R. Meyer, "Lie transform tutorial–II," to appear.

[55] W. Nichols and B. Weisfeiler, "Differential formal groups of J. F. Ritt," *American J. Mathematics*, Vol. 104, pp. 943–1003, 1982.

[56] W. Nichols, "The Kostant structure theorems for K/k-Hopf algebras," *J. Algebra*, Vol. 97, pp. 313–328, 1985.

[57] P. Olver, *Applications of Lie Groups to Differential Equations*, Springer-Verlag, New York, 1986.

[58] J. P. Ramis and J. Martinet, "Theórie de Galois différentielle et resommation," *Computer Algebra and Differential Equations*, edited by E. Tournier, Academic Press, San Diego, pp. 117–214, 1989.

[59] R. H. Rand and D. Armbruster, *Perturbation Methods, Bifurcation Theory and Computer Algebra*, Springer-Verlag, New York, 1987.

[60] L. P. Rothschild and E. M. Stein, "Hypoelliptic differential operators and nilpotent groups," *Acta Mathematica*, Vol. 37, pp. 248–315, 1977.

[61] C. Rockland, "Intrinsic nilpotent approximation," *Acta Applicandae Math.*, Vol. 8, pp. 213–270, 1987.

[62] G.-C. Rota, D. Kahaner, and A. Odlyzko, "Finite Operator Calculus," *J. Mathematics and its Applications*, Vol. 42, pp. 685–760, 1973.

[63] W. Schmid, "Poincaré and Lie groups," *Proceedings of Symposia in Pure Mathematics, Vol. 39, Part 1, The Mathematical Heritage of Henri Poincaré*, F. E. Browder, editor, AMS, Providence, pp. 157–168, 1983.

[64] F. Schwarz, "Symmetries of Differential Equations: From Sophus Lie to Computer Algebra," *Siam Review*, Vol. 30, pp. 450–481, 1988.

[65] C. Scovel, "Symplectic Numerical Integration of Hamiltonian Systems," *Proceedings of the MSRI Workshop on the Geometry of Hamiltonian Systems*, to appear.

[66] R. Shtokhamer, N. Glinos and B. F. Canviness, "Computing elementary first integrals of differential equations," to appear.

[67] M. F. Singer, "An outline of differential galois theory," *Computer Algebra and Differential Equations*, edited by E. Tournier, Academic Press, San Diego, pp. 3–57, 1989.

[68] S. Steinberg and P. J. Roache, "Using Macsyma to Write Fortran Subroutines," *Journal of Symbolic Computation*, Vol. 2, pp. 213–228, 1986.

[69] R. S. Strichartz, "The Campbell-Baker-Hausdorff-Dynkin formula and solutions of differential equations," *Journal of Functional Analysis*, Vol. 72, pp. 320–345, 1987.

[70] M. Takeuchi, "A Hopf algebraic approach to Picard-Vessiot theory," to appear.

[71] G. Viennot and P. Leroux, "A Combinatorial Approach to Nonlinear Functional Expansions: An Introduction with Example," *Algebraic and Computing Treatment of Noncommutative Power Series*, G. Jacob and C. Reutenauer, editors, Theoretical Computer Science, 1990.

[72] P. Wang, "FINGER: A symbolic system for automatic generation of numerical programs in finite element analysis," *Journal of Symbolic Computation*, Vol. 2, pp. 305–324, 1986.

[73] R. Zippel, "Automation of Numerical ODE Solving Techniques," to appear.

P-12

N91-28789

Computations involving differential operators and their actions on functions[*]

Peter Crouch[†]
Arizona State
Robert Grossman[‡] and Richard Larson[§]
University of Illinois at Chicago

December, 1991

Abstract

This extended abstract further develops the algorithms in [8] and [9] for rewriting expressions involving differential operators. The differential operators that we have in mind arise in the local analysis of nonlinear dynamical systems. In this work, we extend these algorithms in two different directions: We generalize the algorithms so that they apply to differential operators on groups and we develop the data structures and algorithms to compute symbolically the action of differential operators on functions. Both of these generalizations are needed for applications.

[*]This is an extended abstract; a final report will be published elsewhere.
[†]This research was supported in part by the NSF grant INT-8914643.
[‡]This research is supported in part by NASA grant NAG2-513 and NSF Grant DMS-8904740.
[§]This research is supported by NSF Grant DMS-8904740.

1 Introduction

This extended abstract further develops the algorithms in [8] and [9] for rewriting expressions involving differential operators. The differential operators that we have in mind arise in the local analysis of nonlinear dynamical systems. In this work, we extend these algorithms in two different directions:

1. We generalize the algorithms so that they apply to differential operators on groups. This generalization is important for applications. For example, the nonlinear system describing a robotic joint or a satellite evolves on the group $G = SO(3)$ of spatial rotations. The local study of such systems requires the computation of expressions consisting of differential operators on G.

2. We develop the data structures and algorithms to compute symbolically the action of differential operators on functions. Again, this is crucial for applications. We illustrate this by deriving conditions for a numerical algorithm to remain constrained to a group. In other words, if $x_{n+1} = T(x_n)$ is the update rule for a numerical algorithm evolving on a group G, we would like to choose T so that $x_n \in G$ implies $x_{n+1} \in G$.

For a further discussion of applications of these algorithms, see [5] and [6] and the references given there.

Here is the

Set Up.

1. Let k denote either the real or complex numbers.

2. Let G denote a finite dimensional Lie group over k, g denote its Lie algebra, and $Y_1, \ldots, Y_N$ a basis for g of left-invariant vector fields.

3. Let $R = C^\infty(G)$ denote the algebra of smooth functions on G taking values in k.

4. Fix M derivations of R of the form

$$F_j = \sum_{\mu=1}^{N} a_j^\mu Y_\mu, \quad a_j^\mu \in R, \quad j = 1, \ldots, M, \tag{1}$$

   and let A denote the free associative algebra $k<F_1, \ldots, F_M>$ of differential operators generated by $F_1, \ldots, F_M$, with coefficients from k.

We are concerned with the following

Problem. Given a differential operator $p \in A$ and a function $f \in R$, substitute the Equations (1) and compute $p \cdot f$ using as few operations as possible. This problem is interesting since in many cases cancellations take place.

Example 1. Let $G = \mathbf{R}^N$ denote the abelian group,

$$Y_j = \frac{\partial}{\partial x_j}, \quad j = 1, \ldots, N,$$

the (left invariant) coordinate vector fields, and F_1, F_2, F_3 three fixed vector fields defined in terms of the Y_μ via Equations (1). Then the naive substitution of (1) and simplification of $p \cdot f$, where

$$p = F_3F_2F_1 - F_3F_1F_2 - F_2F_1F_3 + F_1F_2F_3 \in A, \quad f \in R,$$

yields $24N^3$ terms, while more specialized algorithms need only compute the $6N^3$ terms which don't cancel. These types of examples are considered in [8] and [9].

Example 2. Consider the local analysis of a nonlinear system of the form

$$\dot{x}(t) = F(x(t)), \quad x(0) = x^0 \in G, \tag{2}$$

where

$$F = \sum_{j=1}^{M} u_j F_j.$$

In practice, the u_j are constants, functions of time, or perturbation parameters. The study of this system typically involves the computations of various series in the algebra A of differential operators. For example, the local flow of the system is determined by the Taylor series

$$\exp hF = 1 + hF + \frac{h^2}{2!}F^2 + \frac{h^3}{3!}F^3 + \ldots \in A[[h]].$$

An alternative to computing higher derivatives F^k is to choose constants c_i, c_{ij}, $i = 1, \ldots, k$, $j < i$, so that the expression

$$\exp hc_k\bar{F}_k \cdots \exp hc_1\bar{F}_1,$$

where

$$\begin{aligned}
\bar{F}_1 &= \sum_{\mu=1}^{N} a^\mu(x^0)Y_\mu \in g \\
\bar{F}_2 &= \sum_{\mu=1}^{N} a^\mu(\exp(hc_{21}\bar{F}_1)\cdot x^0)Y_\mu \in g \\
\bar{F}_3 &= \sum_{\mu=1}^{N} a^\mu(\exp(hc_{32}\bar{F}_2)\cdot \exp(hc_{31}\bar{F}_1)\cdot x^0)Y_\mu \in g, \\
&\vdots
\end{aligned}$$

is equal to $\exp hF$ to order k. Notice that the left invariant vector fields $\bar{F}_j$ arise by "freezing the coefficients" of F at various points along its flow.

Expanding these expressions around the common base point $x^0 \in G$ yields many terms, which must cancel in the end if the algorithm is going to approximate the flow of the underlying nonlinear system. The action of the differential operators $\bar{F}_j$ on the coefficient functions a_k^μ must also be computed. Notice, that unlike Example 1, the Y_μ here do not commute. This example will be considered in more detail in Section 4.

The computations in both examples are easily kept track of by using finite rooted trees, labeled with the symbols $F_1, \ldots, F_M$. It turns out the the vector space, with basis the set of such trees, has an algebraic structure B which is crucial to efficiently organizing the computation. The advantage of working with the trees B is that many terms which cancel in the end need not be computed. See [6] for an expository treatment of this idea. The key observation required for this work is that it is possible to define an action of the algebra B of finite rooted trees, labeled with $F_1, \ldots, F_M$, on the ring of functions R which enjoys essentially all the properties of the familiar action of the algebra A of differential operators generated by $F_1, \ldots, F_M$ on R. It turns out that B is a Hopf algebra, just as A is, and that both actions give R the structure of what is called an H-module algebra.

In Section 2, we review the relevant material from algebra. This material may be skimmed on a first reading. In Section 3, we define the Hopf algebra of Cayley trees and its action on the ring of functions R. In Section 4, we continue the discussion of Example 2.

2 H-module algebras

In this section we review the basic facts about bialgebras and H-module algebras which will be used in the remainder of this paper.

In this section, k can be any field of characteristic 0. By an ***algebra*** we mean a vector space A over the field k with an associative multiplication and unit. The multiplication can be represented by a linear map $\mu : A \otimes_k A \to A$; the unit can be represented by a linear map $k \to A$ (the map sending $1 \in k$ to $1 \in A$). The facts that the multiplication is associative, and that $1 \in A$ is a unit, can be expressed by the commutativity of certain diagrams. For example, the commutativity of the diagram

$$\begin{tikzcd}
A \otimes_k A \otimes_k A \arrow[r] \arrow[d] & A \otimes_k A \arrow[d] \\
A \otimes_k A \arrow[r] & A
\end{tikzcd}$$

where the upper horizontal arrow is the map $\mu \otimes I$, the left vertical arrow is the map $I \otimes \mu$, and the remaining two arrows are the map μ, expresses the associativity of multiplication.

The dual notion to an algebra is a ***coalgebra***: a vector space C over the field k together with a coassociative coproduct $\Delta : C \to C \otimes_k C$ and a counit $\epsilon : C \to k$. The fact that Δ is coassociative and that ϵ is a counit is expressed by diagrams which are dual to the diagrams which express the facts that the multiplication of an algebra is associative, and that $1 \in A$ is a unit: they are the same diagrams, with the direction of all arrows reversed. For example, coassociativity is expressed by the commutativity of the diagram

$$\begin{tikzcd}
C \otimes_k C \otimes_k C & C \otimes_k C \arrow[l] \\
C \otimes_k C \arrow[u] & C \arrow[l] \arrow[u]
\end{tikzcd}$$

where the upper horizontal arrow is the map $\Delta \otimes I$, the left vertical arrow is the map $I \otimes \Delta$, and the remaining two arrows are the map Δ. Often the element $\Delta(c) \in C \otimes_k C$ is written $\sum_{(c)} c_{(1)} \otimes c_{(2)}$.

A ***bialgebra*** is a vector space H over k which has both an algebra and a coalgebra structure, such that the coalgebra structure maps are algebra homomorphisms, or equivalently, the algebra structure maps are coalgebra homomorphisms. (This equivalence can be seen by expressing the assertion that the coalgebra structure maps are algebra homomorphisms as a set of commutative diagrams: this set of diagrams is self-dual.)

Some examples of bialgebras are the following:

1. Let G be a group, and let kG be the group algebra of G: the vector space kG has the elements of G as a basis, with multiplication defined by extending the multiplication on G linearly. The coproduct and counit of kG are defined by

$$\left.\begin{array}{rcl}\Delta(g) & = & g\otimes g \\ \epsilon(g) & = & 1\end{array}\right\} \qquad g\in G.$$

2. Let G be an affine algebraic group, and let $k[G]$ be the algebra of representative functions on G. The algebra structure of $k[G]$ is the usual algebra structure of functions with point-wise multiplications. The coproduct arises from the group multiplication $G\times G\rightarrow G$, which induces the map $k[G]\rightarrow k[G\times G]\cong k[G]\otimes_k k[G]$. The counit arises from the map $\{e\}\rightarrow G$, where $\{e\}$ is the single-element group.

3. Let L be a Lie algebra over k, and let $U(L)$ be the universal enveloping algebra of L. The coproduct and counit of $U(L)$ are defined by

$$\left.\begin{array}{rcl}\Delta(x) & = & 1\otimes x+x\otimes 1 \\ \epsilon(x) & = & 0\end{array}\right\} \qquad x\in L,$$

   and extended to all of $U(L)$ using the fact that Δ and ϵ are algebra homomorphisms.

Usually, in studying bialgebras, an additional condition is imposed which is analogous to the assertion that a semigroup is a group. Such bialgebras are called *Hopf algebras*. The bialgebras which we consider in this paper (such as the universal enveloping algebra of a Lie algebra) satisfy this condition automatically.

A coalgebra is said to be *cocommutative* if it satisfies $\Delta = T\circ\Delta$, where T is the map $T: C\otimes_k C\rightarrow C\otimes_k C$ defined by $T(x\otimes y) = y\otimes x$. Note that the bialgebras in Examples 1 and 3 are cocommutative.

A vector space V over k is said to be ***graded*** if there is a sequence of subspaces V_0, V_1, ... such that

$$V\cong\bigoplus_{n=0}^{\infty}V_n.$$

A graded vector space V is said to be *connected* if $V_0\cong k$.

Let H be a bialgebra. A *H-module algebra* is an algebra R which is an H-module such that the action satisfies

$$h \cdot (fg) = \sum_{(h)} (h_{(1)} \cdot f)(h_{(2)} \cdot g), \qquad \text{for all } h \in H, \quad f, g \in R.$$

Remark 2.1 If $g \in H$ satisfies $\Delta(g) = g \otimes g$ and R is an H-module algebra, then g acts as an endomorphism of R; if $x \in H$ satisfies $\Delta(x) = 1 \otimes x + x \otimes 1$ and R is an H-module algebra, then x acts as a derivation of R.

3 H-module algebras and Cayley trees

In this section we describe a bialgebra structure on the vector space with basis all equivalence classes of rooted trees. The relation between trees and differential operators goes back at least as far as Cayley [3] and [4]. Important use of this relation has been made by Butcher in his work on higher order Runge-Kutta algorithms [1] and [2]. In this section and the next, we follow the treatment in [8] and [9]. By a tree we mean a nonempty finite rooted tree, and by a forest we mean a finite family of finite rooted trees, possibly empty.

Suppose $\{F_1, \ldots, F_M\}$ is a set of formal symbols (which later will be the names of differential operators). By a *labeled tree* we mean a tree for which we have assigned an element of $\{F_1, \ldots, F_M\}$ to each node, other than the root, of the tree. We say that a tree is *ordered* in case there is a partial ordering on the nodes such that the children of each node are non-decreasing with respect to the ordering.

We now describe the bialgebra structure on spaces of trees. Let

$$k\{\mathcal{T}(F_1, \ldots, F_M)\}$$

denote the vector space which has as basis all equivalence classes of labeled, ordered trees. The vector space $k\{\mathcal{T}(F_1, \ldots, F_M)\}$ is graded, with the grading given as follows: if the tree t has $n+1$ nodes, then

$$t \in k\{\mathcal{T}(F_1, \ldots, F_M)\}_n.$$

We now define the multiplication on $k\{\mathcal{T}(F_1, \ldots, F_M)\}$. Since the set of labeled, ordered trees form a basis for $k\{\mathcal{T}(F_1, \ldots, F_M)\}$, it is sufficient to describe the product of two such trees. Suppose that t_1 and t_2 are labeled, ordered 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 all the original labels. Order the nodes in the product so that the nodes which originally belonged to each tree retain the same relative order to each other, but all the nodes that orginally belonged to t_1 are greater in the ordering than the nodes that originally belonged to t_2. 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 trivial labeled tree consisting only of the (unlabeled) root is a right and left unit for this product. For details, see [7].

We now define the comultiplication on $k\{\mathcal{T}(F_1, \ldots, F_M)\}$. If t is a tree whose root has children $s_1, \ldots, s_r$, the coproduct $\Delta(t)$ is the sum of the 2^r terms $t_1 \odot t_2$, where the children of the root of t_1 and the children of the root of t_2 range over all 2^r possible partitions of the children of the root of t into two subsets. The labels remain the same, and the ordering is handled in the same way as in the product. The map ϵ which sends the trivial labeled tree to 1 and every other tree to 0 is a counit for this coproduct. In [7], it is shown that these algebra and coalgebra structures are compatible, proving the

Theorem 3.1 *The space $k\{\mathcal{T}(F_1, \ldots, F_M)\}$ is a graded connected cocommutative bialgebra.*

We call this algebra the algebra of *Cayley trees*.

We now define an action of the algebra of Cayley trees

$$B = k\{\mathcal{T}(F_1, \ldots, F_M)\}$$

on the ring R, making R a B-module algebra, which captures the action of trees as higher derivations. The action is defined using the map

$$\psi : k\{\mathcal{T}(F_1, \ldots, F_M)\} \to \mathrm{End}_k R,$$

as follows:

1. Given a labeled, ordered tree t with $m+1$ nodes, assign the root the number 0 and assign the remaining nodes the numbers $1, \ldots, m$. We identify the node with,the number assigned to it. To the node k associate the summation index μ_k. Denote $(\mu_1, \ldots, \mu_m)$ by μ.

2. For the labeled tree t, let k be a node of t, labeled with F_{γ_k} if $k > 0$, and let $l, \ldots, l'$ be the children of k. Define

$$\begin{aligned} R(k;\mu) &= Y_{\mu_l} \cdots Y_{\mu_{l'}} a^{\mu_k}_{\gamma_k}, \quad \text{if } k > 0 \text{ is not the root;} \\ &= Y_{\mu_l} \cdots Y_{\mu_{l'}}, \quad \text{if } k = 0 \text{ is the root.} \end{aligned}$$

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

3. Define

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

4. Extend ψ to all of $k\{\mathcal{T}(F_1, \ldots, F_M)\}$ by linearity.

It is straightforward to check that this action of B on R makes R into a B-module algebra.

We summarize with the following theorem.

Theorem 3.2 *Let G denote a finite dimensional Lie group and R the algebra of smooth functions on G, as detailed in the Set Up. Let B denote the algebra of Cayley trees $k\{\mathcal{T}(F_1, \ldots, F_M)\}$. Then R is a B-module algebra with respect to the action defined by ψ.*

Remark 3.1 The standard action of the algebra A of differential operators generated by $F_1, \ldots, F_M$ on the algebra of smooth functions R gives R the structure of a A-module algebra. It is easy to relate these two H-module algebra structures on R and this observation is the basis for our algorithms.

Let

$$\phi : A \longrightarrow B$$

denote the map sending the generator F_j of the algebra A to the tree consisting of two nodes: the root and a single child labeled F_j. Extend ϕ to be an algebra homomorphism. Let χ denote the map

$$A \rightarrow \mathrm{End}_k R$$

defined by using the substitution (1) and simplifying to obtain an endomorphim of R.

Theorem 3.3 *(i) The maps χ, ϕ and ψ are related by $\chi = \psi \circ \phi$. (ii) Fix a function $f \in R$ and a differential operator $p \in A$. Then*

$$p \cdot f = \phi(p) \cdot f.$$

Here the action on the left views R as an A-module algebra, while the action on the right views R as B-module algebra.

The first assertion is proved in [9] and the second assertion follows from the first assertion and the definitions.

Using this theorem, it is easy to give an algorithm to solve the Problem posed in Section 1. We defer to later paper a complete analysis of the complexity of the algorithm and simply remark here that in many examples the algorithm results in a savings which is exponential in the degree of the differential operator.

Algorithm. Given a smooth function $f \in R$ and a differential operator $p \in A$, compute the function $p \cdot f$ via $\phi(p) \cdot f$.

4 Applications

We use the notation of the Set Up from Section 1. Let $\exp(hF)x$ denote the resulting of flowing for time h along the trajectory of the nonlinear system (2) through the initial point $x^0 \in G$. We require a theorem concerned with the explicit computation of terms in the Taylor series expansion of a solution of (2). This is one of the main applications of the symbolic calculus described in the sections above.

This theorem is most easily stated if we introduce two additional operations on the algebra of Cayley trees B. Given $\alpha, \beta \in B$, define the *meld product* $\beta \odot \alpha$ to be the labeled, ordered tree obtained by identifying the roots of the two trees. The meld product is then extended to all of B by linearity. Given a derivation $F \in \mathrm{Der}(R)$, let β be the tree $\phi(F)$ and let $\alpha \in B$. Recall β is a tree consisting of a root and a node labeled F. We define the *composition product* $\beta \circ \alpha$ to be the tree formed by attaching the subtrees whose roots are the children of the root of α to the node labeled F of the tree β. If $\alpha \in B$ is a tree, define the *exponential* and *Meld-exponential* of a tree by the formal power series

$$\exp(h\alpha) = 1 + h\alpha + \frac{h^2}{2!}\alpha^2 + \frac{h^3}{3!}\alpha^3 + \cdots$$

$$\mathrm{Mexp}(h\alpha) = 1 + h\alpha + \frac{h^2}{2!}\alpha \odot \alpha + \frac{h^3}{3!}\alpha \odot \alpha \odot \alpha + \cdots.$$

Theorem 4.1 *(i) Assume $f \in R$ and $F \in \mathrm{Der}(R)$. If f is analytic near x, then for sufficiently small h,*

$$f(\exp(hF)x) = \exp(h\phi(F)) \cdot f|_x.$$

(ii) Let $F = \sum_{\mu=1}^{N} a^{\mu}(\exp(hG)x^0)Y_\mu$, *where* $G \in \mathrm{Der}(R)$, *and* $x^0 \in G$. *Then*

$$F \cdot f = (\phi(F) \circ \mathrm{Mexp}(hG)) \cdot f.$$

Using this theorem, it is easy to analyze the numerical algorithm described in Example 2 of Section 1. For typographical reasons, we use the following one dimensional notation for trees[1]: the tree consisting of a root and a single child labeled F_1 is denoted $I[F_1]$; the tree consisting of a root and two children labeled F_1 and F_2 is denoted $I[F_1, F_2]$; the tree consisting of a root, with a single child labeled F_1, which itself has two children labeled F_2 and F_3 is denoted $I[F_1[F_2, F_3]]$, etc. Note that the labels need not be distinct, but their order is important.

Consider the expression

$$\exp hc_3F_3 \exp hc_2F_2 \exp hc_1F_1$$

computed to order h^3. Let $p \in A$ denote the resulting expression. The image $\phi(p) \in B$ contains the following terms:

$$\frac{h^3c_2c_{21}^2}{2!}I[F[F,F]] + \frac{h^3c_3c_{31}^2}{2!}I[F[F,F]] + \frac{h^3c_3c_{32}^2}{2!}I[F[F,F]]$$
$$+h^3c_3c_{31}c_{32}I[F[F,F]].$$

Our goal is to choose the constants c_i and c_{ij} so that that

$$\exp hF = p + O(h^4).$$

One of the third order term arising from $\phi(\exp hF)$ is $\frac{h^3}{3!}I[F,[F,F]]$. Setting the coefficients of these trees equal to each other yields the constraint:

$$\frac{c_2c_{21}^2}{2!} + \frac{c_3c_{31}^2}{2!} + \frac{c_3c_{32}^2}{2!} + c_3c_{31}c_{32} = \frac{1}{3!}$$

Other constraints arise from the other trees. We have coded this algorithm in Maple, Mathematica, and Snobol and are currently experimenting with it.

[1]The notation is due to Peter Olver, as is some of the Mathematica code used to generate these examples.

References

[1] J. C. Butcher, "An order bound for Runge-Kutta methods," *SIAM J. Numerical Analysis*, **12** (1975), pp. 304–315.

[2] J. C. Butcher, *The Numerical Analysis of Ordinary Differential Equations*, John Wiley, 1986.

[3] A. Cayley, "On the theory of the analytical forms called trees," in *Collected Mathematical Papers of Arthur Cayley*, Cambridge Univ. Press, Cambridge, 1890, Vol. 3, pp. 242–246.

[4] A. Cayley, "On the analytical forms called trees. Second part," in *Collected Mathematical Papers of Arthur Cayley*, Cambridge Univ. Press, Cambridge, 1891, Vol. 4, pp. 112–115.

[5] R. Grossman, "The evaluation of expresssions involving higher order derivations," *Journal of Mathematical Systems, Estimation, and Control*, Vol. 1, 1990.

[6] R. Grossman, "Using trees to compute approximate solutions of ordinary differential equations exactly," *Computer Algebra and Differential Equations*, M. F. Singer, editor, Academic Press, New York, 1991, in press.

[7] R. Grossman and R. Larson, "Hopf algebraic structures of families of trees," *J. Algebra*, Vol. 26 (1989), pp. 184–210.

[8] R. Grossman and R. Larson, "Labeled trees and the efficient computation of derivations," in *Proceedings of 1989 International Symposium on Symbolic and Algebraic Computation*, ACM, 1989, pp. 74-80.

[9] R. Grossman and R. Larson, "The symbolic computation of derivations using labeled trees," *Journal of Symbolic Computation*, to appear.

N91-28790

Bialgebra deformations and algebras of trees[*]

Robert Grossman[†] and David Radford

Abstract

Let A denote a bialgebra over a field k and let $A_t = A[[t]]$ denote the ring of formal power series with coefficients in A. Assume that A is also isomorphic to a free, associative algebra over k. We give a simple construction which makes A_t a bialgebra deformation of A. In typical applications, A_t is neither commutative nor cocommutative. In the terminology of [1], A_t is a quantum group. This construction yields quantum groups associated with families of trees.

1 Introduction

Let A denote a bialgebra over a field k and let $A_t = A[[t]]$ denote the ring of formal power series with coefficients in A. Assume that A is also isomorphic to a free, associative algebra over k. We give a simple construction which makes A_t a bialgebra deformation of A. In typical applications, A_t is neither commutative nor cocommutative. In the terminology of [1], A_t is a quantum group. This is an extended abstract. The final detailed version of this paper [6] containing proofs and additional examples has been submitted for publication elsewhere.

An interesting class of examples is obtained by taking the bialgebra A to be the Hopf algebra associated with certain families of trees as in [4]. In fact, these examples are closely related to each other and to algorithms pertaining to differential operators [5].

Formal deformations of bialgebras and quantum groups has also been studied from related viewpoints by Drinfeld [1], Gerstenhaber [2], and Gerstenhaber and Schack [3]

[*]This is an extended abstract. The final detailed version of this paper has been submitted for publication elsewhere.

[†]This research is supported in part by NASA grant NAG2-513, NSF Grant DMS-8904740, and by the Laboratory for Advanced Computing at the University of Illinois at Chicago.

2 Power series and the completed tensor product

Suppose that k is a field and A and B are k-algebras. We let $A_t = A[[t]]$ denote the ring of formal power series over A with its usual multiplication. Let $f : A \longrightarrow B_t$ be a k-linear map and write $f(a) = \sum_{n=0}^{\infty} c_f(a,n)t^n$ for $a \in A$. Define a k-linear map $\widehat{f} : A_t \longrightarrow B_t$ by

$$\widehat{f}(\mathbf{a}) = \sum_{n=0}^{\infty}(\sum_{i+j=n} c_f(a_i, j))t^n, \quad \text{for } \mathbf{a} = \sum_{n=0}^{\infty} a_n t^n \in A_t.$$

Observe that we can define a cateogy $(Alg_k)_t$, whose objects are A_t and whose morphisms are k-linear maps $f : A_t \longrightarrow B_t$ satisfying $f = \widehat{f|_A}$, where $f|_A$ is the restriction of f to A.

We define the *completed tensor product* $A_t \widehat{\otimes}_{k_t} B_t$ of A_t and B_t over k_t in this category to be $(A \otimes_k B)_t$. For $\mathbf{a} = \sum_{n=0}^{\infty} a_n t^n \in A_t$ and $\mathbf{b} = \sum_{n=0}^{\infty} b_n t^n \in B_t$ we let

$$\mathbf{a} \widehat{\otimes} \mathbf{b} = \sum_{n=0}^{\infty}(\sum_{i+j=n} a_i \otimes b_j)t^n.$$

Suppose that $f : A_t \longrightarrow A'_t$ and $g : B_t \longrightarrow B'_t$ are morphisms. We define a morphism of completed tensor products $f \widehat{\otimes} g : A_t \widehat{\otimes}_{k_t} B_t \longrightarrow A'_t \widehat{\otimes}_{k_t} B'_t$ by setting $(f \widehat{\otimes} g)|_{A \otimes_k B}(a \otimes b) = f(a) \widehat{\otimes} g(b)$ for $a \in A$ and $b \in B$. The usual formalism for the linear tensor product of maps translates to

$$(f \widehat{\otimes} g)(\mathbf{a} \widehat{\otimes} \mathbf{b}) = f(\mathbf{a}) \widehat{\otimes} g(\mathbf{b})$$

for $\mathbf{a} \in A_t$ and $\mathbf{b} \in B_t$ in this category. Note that if f and g are also algebra maps, then the morphism $f \widehat{\otimes} g : A_t \widehat{\otimes}_{k_t} B_t \longrightarrow A'_t \widehat{\otimes}_{k_t} B'_t$ is an algebra map.

Now let C be any coalgebra over k. A sequence of elements $c_0, c_1, c_2, \ldots \in C$ is called a sequence of divided powers if

$$\Delta(c_n) = \sum_{i+j=n} c_i \otimes c_j \qquad \text{and} \qquad \epsilon(c_n) = \delta_{0,n} \qquad \text{for all } n \geq 0.$$

Probably the most basic example of a sequence of divided powers arises from a primitive element in a bialgebra. Suppose that A is a bialgebra over k and that $\ell \in A$ is primitive. Since Δ is multiplicative, we calculate by the binomial theorem that $\Delta(\ell^n) = (\Delta(\ell))^n = (\ell \otimes 1 + 1 \otimes \ell)^n = \sum_{i=0}^{n}\binom{n}{i}(\ell^{n-i} \otimes \ell^i)$ for $n \geq 0$. Thus $c_0, c_1, c_2, \ldots$ is a sequence of divided powers, where $c_n = \frac{\ell^n}{n!}$ for $n \geq 0$.

3 Deformations of certain enveloping algebras

The notions of algebra, coalgebra, bialgebra and Hopf algebra in the category $(Alg_k)_t$ are the same as those in the category of vector spaces over the field k except, of course, the structure maps are required to be morphisms. Let (A, m, η) be an algebra over k, where $m : A \otimes A \longrightarrow A$ is multiplication and $\eta : k \longrightarrow A$ defines the unity of A. Then $(A_t, \widehat{m}, \widehat{\eta})$ is an algebra in $(Alg_k)_t$. It is easy to see that $\widehat{m}(\mathbf{a}\widehat{\otimes}\mathbf{b}) = \mathbf{ab}$ for $\mathbf{a}, \mathbf{b} \in A_t$. A morphism $f : A_t \longrightarrow B_t$ is a morphism of algebras if and only $f|_A : A \longrightarrow B_t$ is a map of k-algebras.

The proof of the proposition below is really a matter of unravelling definitions.

Proposition 1 *Suppose that A is an algebra over a field k with a k-coalgebra structure (A, Δ, ϵ). Then $(A_t, \widehat{\Delta}, \widehat{\epsilon})$ is a coalgebra in $(Alg_k)_t$. Furthermore*

$$\widehat{\Delta}(\mathbf{a}) = \sum_{n=0}^{\infty} (\Delta(a_n))t^n \qquad \text{and} \qquad \widehat{\epsilon}(\mathbf{a}) = \sum_{n=0}^{\infty} \epsilon(a_n)t^n$$

for $\mathbf{a} = \sum_{n=0}^{\infty} a_n t^n \in A_t$.

Suppose that (A_t, Δ, ϵ) is a coalgebra in $(Alg_k)_t$. We say that $K \in A_t$ is grouplike if

$$\Delta(K) = K\widehat{\otimes}K \qquad \text{and} \qquad \epsilon(K) = 1.$$

We say that $\ell \in A_t$ is nearly primitive if

$$\Delta(\ell) = \ell\widehat{\otimes}K + H\widehat{\otimes}\ell$$

for some grouplike elements $K, H \in A_t$. If $K = H = 1$ then ℓ is said to be primitive.

For an algebra A over a field k of characteristic 0 we let $exp(at) = \sum_{n=0}^{\infty}(\frac{a^n}{n!})t^n \in A_t$. The following corollary gives the relationship between sequences of divided powers and grouplike elements.

Corollary 1 *Suppose that A is an algebra over a field k which has a k-coagebra structure (A, Δ, ϵ). Let $(A_t, \widehat{\Delta}, \widehat{\epsilon})$ be the resulting coalgebra in $(Alg_k)_t$. Then:*

(a) *Let $K = \sum_{n=0}^{\infty} a_n t^n \in A_t$. Then K is grouplike if and only if $a_0, a_1, a_2, \ldots$ is a sequence of divided powers in A.*

(b) *Suppose that the characteristic of k is 0. If $a \in A$ is primitive, then $K = exp(at)$ is a grouplike element of A_t.*

Now we construct deformations of enveloping algebras over a field of characteristic 0 which are free as associative algebras on a space of primitives.

Theorem 1 *Suppose that V is a vector space over a field k of characteristic 0. Turn the tensor algebra (A, m, η) of V into a bialgebra $(A, m, \eta, \Delta, \epsilon)$ by defining $\Delta(\ell) = \ell \otimes 1 + 1 \otimes \ell$ and $\epsilon(\ell) = 0$ for $\ell \in V$. Let $p, q \in V$ and write V as a direct sum of subspaces $V = P \oplus P'$, where $P = span(p, q)$. Then there is bialgebra deformation $(A_t, \widehat{m}, \widehat{\eta}, \widehat{\boldsymbol{\Delta}}, \widehat{\epsilon})$ of $(A, m, \eta, \Delta, \epsilon)$ such that*

a) $\widehat{\boldsymbol{\Delta}}(\ell) = \ell \widehat{\otimes} 1 + 1 \widehat{\otimes} \ell$ *for* $\ell \in P$,

b) $K = exp(pt)$ *and* $H = exp(qt)$ *are grouplike elements of* $(A_t, \widehat{\boldsymbol{\Delta}}, \widehat{\epsilon})$, *and*

c) $\widehat{\boldsymbol{\Delta}}(\ell) = \ell \widehat{\otimes} K + H \widehat{\otimes} \ell$ *for* $\ell \in P'$.

We comment that $K = exp(tp) = 1$ when $p = 0$. If $p \neq 0$ and $q = 0$, for example, then the deformation of the theorem is not cocommutative. If $dim(V) > 1$ then the free algebra A is not commutative. In this case the deformation of the theorem is not commutative.

4 Deformations of bialgebras trees

In this section we give an example from [4] involving bialgebras of trees to which Theorem 1 applies. Let k be a field of characteristic 0. Let $\mathcal{T}$ be the set of finite rooted trees, and let $k\{\mathcal{T}\}$ be the k-vector space which has $\mathcal{T}$ as a basis.

We first define a coalgebra structure on $k\{\mathcal{T}\}$. If $t \in \mathcal{T}$ is a tree whose root has children $s_1, \ldots, s_r$, the coproduct $\Delta(t)$ is the sum of the 2^r terms $t_1 \otimes t_2$, where the children of the root of t_1 and the children of the root of t_2 range over all 2^r possible partitions of the children of the root of t into two subsets. The map ϵ which sends the trivial tree to 1 and every other tree to 0 is a counit for this coproduct. It is easy to see that comultiplication is cocommutative.

We now define an algebra structure on $k\{\mathcal{T}\}$. Suppose that $t_1, t_2 \in \mathcal{T}$ are 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 tree t_2 by making each s_i the child of some node of t_2. The product $t_1 t_2$ is defined to be the sum of these $(n+1)^r$ trees. It turns out that this product is associative, and that the trivial tree e consisting only of the root is a right and left unit. It can also be shown that the maps defining the coalgebra structure are algebra homomorphisms, so that $A = k\{T\}$ is a bialgebra.

For technical reasons, we require that nodes of the tree be ordered. We say that a rooted finite tree is *ordered* in case there is a partial ordering on the nodes such that the children of each node are non-decreasing with respect to the ordering. To define the product of two ordered trees t_1 and t_2, compute the product of the underlying trees, and order the nodes in the product so that the nodes which originally belonged to each tree retain the same relative order to each other, but all the nodes that orginally belonged to t_1 are greater in the ordering than the nodes that originally belonged to t_2.

Let X be the set of of trees whose root has one child. Then $A \cong k{<}I(X){>}$ as an algebra [4]. Applying Theorem 1 yields a bialgebra deformation A_t of A which is neither commutative nor cocommutative.

References

[1] V. G. Drinfeld, "Quantum Groups," *Proceedings of the International Congress of Mathematicians*, Berkeley, California, pp. 798–819, 1987.

[2] M. Gerstenhaber, "On the deformations of rings and algebras," *Annals of Mathematics*, Vol. 79, pp. 59–103, 1964.

[3] M. Gerstenhaber and S. D. Schack, "Bialgebra cohomology, deformations, and quantum groups," *Proceedings of the National Academy of Sciences*, Vol. 87, pp. 478–481, 1990.

[4] R. Grossman and R. G. Larson, "Hopf algebraic structures of families of trees," *J. Algebra*, Vol. 26, pp. 184–210, 1989.

[5] R. Grossman and R. G. Larson, "The symbolic computation of derivations using labeled trees," *Journal of Symbolic Computation*, to appear.

[6] R. Grossman and D. Radford, "Bialgebra deformations of certain universal enveloping algebras," submitted for publication.

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

Address for Robert Grossman: Department of Mathematics, Statistics, and Computer Science, University of Illinois at Chicago, Mail Code 249, Box 4348, Chicago, IL 60680, grossman@uicbert.eecs.uic.edu.

Address for David Radford: Department of Mathematics, Statistics, and Computer Science, University of Illinois at Chicago, Mail Code 249, Box 4348, Chicago, IL 60680, U21523@uicvm.uic.edu.