JetBundleGeometry

`Domain: JBG`

Kurt Pagani
nilqed@gmail.com

December 20, 2016

Abstract

This manual decribes the FriCAS domain **JetBundleGeometry**. This domain extends the JET series and provides methods to compute **pull backs**, **integrals** and other quantities of differential forms living on a Jet bundle. The domain relies on `JET` ([8]) and some other domains and packages.

Contents

1 Introduction

Recall that a bundle is a triple (M, X, π), where M, X are manifolds and $\pi : M \to X$ is the projection, i.e. a continuous surjective mapping from the total space (M) to the base space (X). A bundle is called trivial if M is homeomorphic to $X \times U$, where U denotes another manifold, called the fibre. A locally trivial bundle is called a fibre bundle, which is what we here are concerned with. We will use the following nomenclature:

$$x = (x_1, \ldots, x_p), \quad u = (u^1, \ldots, u^n)$$

where x are the (local) coordinates of the base manifold X and u the the ones on the fibre U. Locally the projection takes the form

$$\pi : \begin{cases} X \times U \\ (x, u) \mapsto x. \end{cases}$$

A section of the fibre bundle then has the form

$$\Phi_f : \begin{cases} X \times U \\ x \mapsto (x, f(x)). \end{cases}$$

such that $\pi \circ \Phi_f = \mathcal{I}d$. A jet bundle now may be considered as an iteration of that construction, that is we will speak of a n-th order jet bundle $M^{(n)}$ if

$$M^{(n)} = X \times (U \times U_1 \times \ldots \times U_n)$$

where the U_j are the jet (Euclidean) spaces. We will just write M for $M^{(0)}$.

1.1 FriCAS JET

JET in FriCAS is a sophisticated series of domains and packages written by Joachim Schue and Werner M. Seiler ([3],[4], [5],[8]). Everything is contained in the source file `jet.spad` (7000+ loc). At the moment JET comprises the following types or packages:

1 `CartanKuranishi` is a package for the completion of a given differential equation to an involutive equation. Procedures for Cartan characters and Hilbert polynomial are also provided. Based on the *Cartan-Kuranishi* theorem as it is used in formal theory.

2 `DistributedJetBundlePolynomial` implements polynomials in a distributed representation. The unknows come from a finite list of jet variables. The implementation is basically a copy of the one of *GeneralDistributedMultivariatePolynomial*.

3 `IndexedJetBundle` provides the standard implementation for a jet bundle with a given number of dependent and independent variables where the variables are given as symbols with upper bounds on the indexes (that's why the prefix *Indexed*. Otherwise it is the same as `JetBundle`.

4 `JetBundle` implements a jet bundle of arbitrary order with given names for the independent and dependent variables. It supports only repeated index notation.

5 `JetBundleBaseFunctionCategory` defines the category of functions (local sections) of the base space of a jet bundle, i.e. functions depending only on the independent variables. Such a category is needed e.g. for the representation of solutions.

6 `JetBundleCategory` provides basic data structures and procedures for jet bundles. Nearly all necessary functions are implemented already here. Only the representation and functions which direct access to it must be implemented in a domain. Two notations of derivatives are supported. Default is multi-index notation, where the i-th entry of the index denotes the number of differentiations taken with respect to x^i. In repeated index notation each entry i in the index denotes a differentiation with respect to x^i. The choice affects, however, only in- and output. Internally, multi-index notation is used throughout.

7 `JetBundleExpression` defines expressions over a jet bundle based on Expression Integer. It allows all kind of algebraic operations. simplify is implemented using Groebner bases in polynomials over kernels. Thus it might not work correctly for general expressions. This also affects dimension.

8 `JetBundleFunctionCategory` defines the category of functions (local sections) over a jet bundle. The formal derivative is defined already here. It uses the Jacobi matrix of the functions. The columns of the matrices are enumerated by jet variables. Thus they are represented as a Record of the matrix and a list of the jet variables. Several simplification routines are implemented already here.

9 `JetBundleLinearFunction` implements linear functions over a jet bundle. The coefficients are functions of the independent variables only.

10 `JetBundlePolynomial` implements polynomial sections over a jet bundle. The order is not fixed, thus jet variables of any order can appear.

11 `JetBundleSymAna` is only necessary to have a valid return type for some procedures in SymmetryAnalysis. It is essentially identical with JetBundle but computes its parameters in a more complicated way.

12 `JetBundleXExpression` implements arbitrary functions in a jet bundle which depend only on the independent variables x. Otherwise it is identical with `JetBundleExpression`. Such a domain is needed for `JetLinearFunction`.

13 `JetCoordinateTransformation` implements changes of local coordinates. Given are the changes of the coordinates of the base space, i.e. the independent and dependent variables. The transformations of the derivatives are computed via the chain rule. $Y(W)$ contains expressions for the old variables in terms of the new ones.

14 `JetDifferential` implements differentials (one-forms) over a jet bundle.The differentials operate on `JetVectorField`.

15 `JetDifferentialEquation` provides the basic data structures and procedures for differential equations as needed in the geometric theory. Differential equation means here always a submanifold in the jet bundle. The concrete equations which define this submanifold are called system. In an object of the type JetDifferentialEquation much more than only the system is stored. D denotes the class of functions allowed as equations. It is assumed that the simplify procedure of D returns only independent equations and a system with symbol in row echelon form.

16 `JetGroebner` provides a procedure to compute *Groebner* bases for arbitrary domains of jet polynomials. Two internal procedures transform to and from DistributedJetBundlePolynomial where the actual computation is done. The argument LJV contains all jet variables effectively occurring in the polynomials. The ordering is determined by the ordering in P.

17 `JetLazyFunction` takes as argument a domain in JetBundleFunctionCategory and returns another domain in the same category. This domain has basically the same properties as the argument domain, but there is a lazy evaluation mechanism for derivatives. This means that differentiations are not immediately performed. Instead a pointer is established to the function to be differentiated. Only when the exact value of the derivative is needed, the differentiation is executed. Special care is taken for leading derivatives and jet variables to avoid as much as possible the need to evaluate expressions. This entails that the result of *jetVariables* may contain spurious variables. Furthermore many functions in *JetLazyFunction* destructively change their arguments. This affects, however, only their internal representation, not the value obtained after full evaluation.

18 `JetVectorField` implements vector fields over the jet bundle JB with coefficients from D. The fields operate on functions from D.

19 `LUDecomposition` contains procedures to solve linear systems of equations or to compute inverses using a `LU` decomposition.

20 `SparseEchelonMatrix` implements sparse matrices whose columns are enumerated by the `OrderedSet` and whose entries belong to a `Gcd` domain. The basic operation of this domain is the computation of an row echelon form. The used algorithm tries to maintain the sparsity and is especially adapted to matrices who are already close to a row echelon form.

21 `SymmetryAnalysis` provides procedures for the symmetry analysis of differential equations over a given jet bundle. Currently there exist only some procedures to set up the determining system for the symmetry generators of *Lie* point symmetries.

The exported functions and how all plays together is best documented in [4]. There you will also find comprehensive implementation notes as well as some examples.

1.2 Motivation

Let us recall the following domains/packages and their functionality:

1 `DeRhamComplex` provides differential forms as a graded ring in a given set of variables:

$$\mathtt{DeRhamComplex}(\mathtt{R}, [\mathtt{x}_1, \ldots, \mathtt{x}_n]) = \bigoplus_{p=0}^{n} \Lambda^p([x_1, \ldots, x_n])$$

where $x_j \in \mathtt{Expression}(\mathtt{R})$, R a ring, meaning that the coefficients are from `Expression R`.

2 `DifferentialForms` is a package extending `DeRhamComplex` by *Riemannian metrics* g and the connected standard operations like *Hodge star* $\star_g$, δ_g, Δ_g, i_X, $\mathcal{L}_X$ and $\langle \cdot, \cdot \rangle_g$.

3 `CellMap` and `SurfaceComplex` are domains which loosely spoken will provide parametric surface patches and their formal sums (free group with integer coefficients):

$$\mathtt{SurfaceComplex}(\mathtt{R}, \mathtt{n}) = \mathbb{Z} \cdot \bigoplus_{p=0}^{n} \Sigma^p([\$_1, \ldots, \$_n])$$

where Σ_p is the collection of $p-$cells, that is mappings from a $p-$cell into $n-$space.

Consequently we can define the boundary operator ∂ such that $T(d\varphi) = \partial T(\varphi)$ holds, where T is an element of SurfaceComplex and φ one of DeRhamComplex.

But note that the code of these domains is unrelated, that is both can be used independently. So we are in need of a package or domain which brings them together in order to utilize their mutual duality.

A second point is the need of computing *pull backs* of differential forms and - closely related - of integration of forms over *affine chains*. To achieve this we have to incorporate two (usually) different spaces of differential forms.

Eventually, the following quotations from [4] together with the remarks above induced the usage of `JetBundle` instead of creating a new domain:

> ...The implementation of both is somewhat rudimentary, as we hope that some day `AXIOM` will contain a reasonable environment for differential geometric calculations and then it should be used instead of some special domains like the two mentioned. ...The implementation of the differential geometric domains `VectorField` (abbreviated by `VF`) and `Differential` `(DIFF)` are fairly primitive. They should be considered as a temporary hack, until AXIOM contains a better environment for differential geometric calculations. ...

We may interpret `JetBundle` in the first place as a local coordinate patch (chart) with independent variables x and dependent ones u. Then we will be able to pull forms in $u-$ space back to x-space. In other words we will only use a zero order jet bundle for the moment. In a second step the interplay might be extended to higher order bundles such that the deficiencies mentioned in the above quotes may be mitigated.

2 Appendix

The following is a verbatim inclusion of the extended abstract of W.M. Seiler's paper.

JET — An AXIOM Environment for Geometric Computations with Differential Equations

W.M. Seiler and J. Calmet

Institut für Algorithmen und Kognitive Systeme

Universität Karlsruhe
76128 Karlsruhe, Germany
Email: seilerw@ira.uka.de

Extended Abstract

Geometric methods play an important role in the analysis of nonlinear differential equations. For example, symmetry methods provide the more or less only systematic approach to the construction of solutions. However, most geometric computations tend to be very tedious. Thus the use of computer algebra systems considerably helps in the application of these methods.

JET is an environment within the computer algebra system AXIOM to perform such computations. The current implementation emphasizes the two key concepts involution and symmetry. It provides some packages for the completion of a given system of differential equations to an equivalent involutive one based on the Cartan-Kuranishi theorem and for setting up the determining equations for classical and non-classical point symmetries.

We stress that JET is an *environment* for such computations and not simply a collection of some special purpose algorithms. Thus it contains general data structures for the jet bundle formalism which can also be used for other tasks than the two above mentioned. Using the generic programming facilities of AXIOM it is possible to provide several representations for jet bundles and for different classes of differential equations. The main application packages are independent of such details.

Involution has important applications in symmetry theory. One should e.g. mention that involutive systems are locally solvable and only for such systems the two widely used definitions of a symmetry coincide. The in calculations applied definition as a transformation leaving the differential equation invariant yields for not locally solvable systems usually less symmetries then the definition as a transformation mapping solutions into solutions. This can easily be seen with the system $u_z + yu_x = u_y = 0$. Obviously it is not invariant under y-translations, but its solution space $u = const$ has this symmetry. This effect has especially implications on the nonclassical method of Bluman and Cole.

Other applications include computing the size of the symmetry group of a differential equation without solving the determining equations. Furthermore it is possible to "correct" the result by subtracting some unwanted effects like e.g. the trivial superposition symmetry of linear equations. In the case of gauge theory the concept of involution leads to an new intrinsic definition for the number of degrees of freedom based on a similar formal correction.

A brief description of an earlier version of JET can be found in Ref. [3]. The current version is described in much detail in Ref. [4]. For more in-

formation about the underlying mathematical theory we refer to Ref. [5]. Applications of the concept of involution in symmetry theory are discussed in Ref. [6]. Finally we mention Ref. [7] for some applications in physics. All these publications can be obtained via WWW at the URL:
`http://www.mathematik.uni-kassel.de/~seiler/`. [1]

References

[1] Singer, I. M., and Thorpe, J. A.*Lecture Notes on Elementary Topology and Geometry*, Scott, Foresman and Company.

[2] Spivak, M. *Calculus on Manifolds*, W. A. Benjamin, Inc., New York, 1965.

[3] J. Schü, W.M. Seiler, and J. Calmet. Algorithmic methods for Lie pseudogroups. In N. Ibragimov, M. Torrisi, and A. Valenti, editors, *Proc. Modern Group Analysis: Advanced Analytical and Computational Methods in Mathematical Physics*, pages 337–344, Acireale (Italy), 1992. Kluwer, Dordrecht 1993.

[4] W.M. Seiler. Applying AXIOM to partial differential equations. Internal Report 95-17, Universität Karlsruhe, Fakultät für Informatik, 1995. Available from: `http://citeseerx.ist.psu.edu`

[5] W.M. Seiler. *Analysis and Application of the Formal Theory of Partial Differential Equations.* PhD thesis, School of Physics and Materials, Lancaster University, 1994.

[6] W.M. Seiler. Involution and symmetry reductions. Preprint 1995.

[7] W.M. Seiler and R.W. Tucker. Involution and constrained dynamics I: The Dirac approach. *J. Phys. A*, to appear, 1995.

[8] W.M. Seiler and Jacques Calmet *JET - An AXIOM Environment for Geometric Computations with Differential Equations* Electronic Proc. IMACS Conference on Applications of Computer Algebra, Albuquerque 1995, M. Wester, S. Steinberg, M. Jahn (eds.)

[1] link updated.