Next Generation Computer Algebra Systems
AXIOM and the Scratchpad Concept:
Applications to Research in Algebra

Larry Lambe

Supported by NSF Grant No. CCR-9207241 and the Department of Mathematics, Stockholm University. The author also wishes to express his gratitude to Professor P-C. Ramusino for his hospitality at the Universities of Trent and Milan where a part of this report was prepared.

Presented to the 21^{st} Nordic Congress of Mathematicians, June, 1992, Luleå, Sweden

Abstract

One way in which mathematicians deal with infinite amounts of data is symbolic representation. A simple example is the quadratic equation $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$, a formula which uses symbolic representation to describe the solutions to an infinite class of equations. Most computer algebra systems can deal with polynomials with symbolic coefficients, but what if symbolic exponents are called for (e.g., $1+t^i$)? What if symbolic limits on summations are also called for (e.g., $1+t+...+t^i=\sum_j t^j$)? The "Scratchpad Concept" is a theoretical ideal which allows the implementation of objects at this level of abstraction and beyond in a mathematically consistent way. The AXIOM computer algebra system is an implementation of a major part of the Scratchpad Concept. AXIOM (formerly called Scratchpad) is a language with extensible parameterized types and generic operators which is based on the notions of domains and categories **[Lambe1]**, **[Jenks-Sutor]**. By examining some aspects of the AXIOM system, the Scratchpad Concept will be illustrated. It will be shown how some complex problems in homological algebra were solved through the use of this system.

§1 Introduction

New paradigms are evolving in computer science. There is a thrust towards typed languages in which object oriented methodologies may be conveniently implemented. In mathematical programming, such concepts were proposed in the 1970's by Richard Jenks [**Jenks**] through the theoretical design of MODLISP. Not much later, at IBM Research, Yorktown Heights, NY, Jenks embarked on the implementation of a system which was intended to allow the mathematical programmer a very high level language in which he could, for example, implement only one algorithm for several types of rings if indeed that algorithm was valid for such rings. For example, since the Euclidian algorithm works over any Euclidian domain, one should not have to implement it once for the integers, again for the Gaussian integers, and

yet another time for polynomials in one variable over a field. The theoretical basis for this system was MODLISP and the system was called Scratchpad. Over Scratchpad's more than 20 year history at Yorktown Heights it underwent many design changes and had many contributors (see [**Jenks-Sutor**]). It has allowed a style of mathematical programming that represents the way many mathematicians actually think about the inter-relationships between the sub-areas within mathematics. This style of mathematical programming will be called the *Scratchpad Concept*. A formal definition will not be given. Instead, examples of this style will be highlighted.

In the early 1990's, the Scratchpad Research Project evolved into a system called AXIOM (a trade mark of NAG, Ltd.) and is now commercially available through the offices of NAG. The system consists of an interpreter, a compiler, an extensive library, and a sophisticated graphical user interface. The Scratchpad Concept will be first illustrated abstractly and then through the use of AXIOM. Examples are taken from the speaker's research in homological algebra.

Some well-known mathematical paradigms are followed, but with some very new twists. For example, one might have a need to deal with formulae given recursively in a situation where closed formulae would yield an immediate solution. One method, employed since the beginning of mathematics, is to simply try some symbolic manipulation aided by the calculation of several cases to get a handle on formulae which might then be proven correct by an induction. In our cases, the symbolic manipulation as well as the calculation of special cases was done using AXIOM, but the resulting formulae are almost as complex as the original forms. However, much progress has been made through this process since the resulting closed forms allow the construction of extremely efficient routines to deal with further calculation and manipulation in AXIOM (or other systems) involving the original structures. It is believed that this very high level of "code generation", so convenient in a system like AXIOM, will be one of the most important contributions to modern computational mathematics of the new generation computer algebra systems.

1.1 Procedures With Mathematical Objects as Parameters

Consider matrix row reduction. It can be accomplished by using the three basic operations

$$ero1(i, j) : \text{Interchange rows } i \text{ and } j$$

$ero2(r, i, j)$: Add r times row i to row j

$ero3(r, i)$: Multiply row j by r

Algorithms for matrix row reduction using these operations can easily be implemented in Pascal, for example. One might have

```
MTX = matrix[1..maxRow, 1..maxCol] of Field;
Field = real;
Int = integer;
```

The procedure call might look like

```
rowReducedEchelon(var m : MTX; n: Int; m : Int);
```

But clearly the *ero*'s can be performed over a number of rings and, indeed, sometimes one encounters problems that call for matrix row reduction over one of these other rings. For example, what if $Ring = GF(3, 2)$, the field with 9 elements? Furthermore, if elements are not in a field, but in a principal ideal domain, then what can one do? It would be nice to be able to write

```
rowReducedEchelon(M : Matrix(n,m,R), R : Ring, n, m : Int).
```

Not only is is more convenient for the user, it is also more convenient for the implementer if only one procedure needs to be written for all admissible types.

Here is a segment that gives natural expression of the algorithm:

```
Ring : Category
R : Ring
n, m : Int
M : Matrix(n,m,R)

rowReducedEchelon(M,R,n,m) ==
  if R is a Field then
       Block1
       Block2
            .
            .
  else if R is a EuclidianDomain then
         Block1'
         Block2' ...
```

For another example, suppose that R is some ring and $X = \{1, t_1, t_2, \ldots\}$ is a set of indeterminates. We can form the free R-module on X. It has X as a basis and so consists of all linear combinations $\sum a_i t_i$ where $a_i \in R$ and the sum is finite. Let's denote this mathematical object by

`FreeModule(R,X)`.

We could define an operation on the elements of X and extend it bilinearly to obtain an algebra over R. For example, define $1t_i = t_i$, $t_i 1 = t_i$, and $t_i t_j = t_{i+j}$. Clearly this algebra is isomorphic to the polynomial ring in one variable t. Call the result `Poly(R,t)`.

Notice that `Poly(R,t)` is an extension of `FreeModule(R,X)`. It has all the operations from `FreeModule(R,X)` and an extra one, viz., a multiplication. Notice also that all of this makes sense for any binary operation on X. A set M with an operation

$$* : M \times M \to M$$

that is associative and has an identity element 1 is called a monoid. For any monoid, there is an operation on $FreeModule(R, M)$ which is obtained by extending the multiplication of M bilinearly. The resulting algebra over R is called the monoid ring. Let's denote that algebra by

`MonoidRing(R,M)`.

Going back to our polynomial example, notice that except for cosmetics, `Poly(R,t)` is just `MonoidRing(R,NonNegativeInteger)` where the second parameter denotes the monoid of non-negative integers under addition of integers and identity 0.

If an element of `FreeModule(R,M)` is represented by a list of terms (with addition implied) then a correspondence of the form

$$r_1 m_1 + \ldots + r_k m_k \leftrightarrow [[r_1, m_2], \ldots [r_k, m_k]]$$

has been set up between formal linear combinations and lists of two element lists where the first member of the two element sublist is the coefficient r_i from R and the second element is the basis element m_i from M. Another abstract correspondence can be given by simply choosing a dummy variable t and matching the list of lists above to linear combinations of expressions of the form t^{m_i}. Thus one can also use the representation

$$[[r_1, m_2], \ldots [r_k, m_k]] \leftrightarrow r_1 t^{m_1} + \ldots + r_k t^{m_k}.$$

With this representation, and when the monoid M is the non-negative integers, one clearly gets ordinary polynomial expressions from `MonoidRing(R,M)`. When the monoid M is the full group of integers, one gets "polynomials" with non-negative *or* negative exponents, i.e., Laurent polynomials. To get polynomials in several indeterminates, several approaches can be taken. Given the discussion of polynomials in one variable, it is easily seen that polynomials in two variables are (except for cosmetics) given by

```
MonoidRing(POL,NonNegativeInteger)
```

where `POL=MonoidRing(R,NonNegativeInteger)`. In fact, one can simply iterate this procedure to obtain polynomials in n variables for any $n \geq 1$.

For another representation, let NNI stand for the non-negative integers with its additive monoid operation. The cartesian product, NNI^n, of n copies of NNI, can be given an induced monoid operation in the usual way by adding coordinate-wise. Denote NNI^n by

```
DirectProduct(n,NNI).
```

The polynomial ring in n indeterminates (except for cosmetics) is then given also by

```
MonoidRing(R,DirectProduct(n,NNI)).
```

We will come back to the cosmetic issues later.

1.3 Skew Symmetric Polynomials

The exterior (or Grassman) algebra is another useful construction. The exterior algebra can be constructed from the free R module with basis in bijective correspondence with the set of all subsets of $\{1, 2, \ldots, n\}$. Think of a basis element as $e_{i_1} \wedge \ldots \wedge e_{i_k}$.

So there is an algebra structure on the module $FreeModule(R, P(X))$ where $P(X)$ is the set of all subsets of X. Thus here is another example of an algebra constructed from $FreeModule(R, Y)$ (for $Y = P(X)$) by *adding on operations*. The new algebra constructor will be called $AntiSymm(R, X)$. This algebra is commonly denoted by $E[e_1, \ldots, e_n]$ and this notation will be used in what follows.

The organization of the construction of the ring of antisymmetric polynomials might be accomplished by a scheme like

$$\text{AntiSymm(R,X)} : \text{Algebra(R)} ==$$

FreeModule(R,P(X)) add

.

.

Think of an element of $AntiSymm(R, X)$ as having "type" $AntiSymm(R, X)$. Think of $AntiSymm(R, X)$ as having "type" $Algebra(R)$. Note that, in this sense, the "category" of algebras is parameterized by rings R.

§2. The AXIOM Language

The AXIOM language allows "categories" and "domains" to be constructed in a natural way.

2.1 Categories, Domains, and Packages

To understand what categories are in AXIOM, think of groups. All groups have a binary operation $*$, a unary operation inv, and a constant 1

$$* \quad : \quad G \times G \longrightarrow G \tag{1}$$

$$inv \quad : \quad G \longrightarrow G \tag{2}$$

$$1 \quad \in \quad G \tag{3}$$

which satisfy certain axioms. Roughly speaking, in the AXIOM language, the specification of the range and domain of a function as given for $*$ and inv above is called a "signature". Also speaking roughly, a category in AXIOM is a collection of signatures for potential domains which are to be of that category. Every domain of a given category should implement each of the signatures in the category, but might contain additional signatures. An example of how one might give the category of groups in AXIOM is given below. (See [**Jenks-Sutor**] for precise definitions of category, domain, and package.)

In the following example, the first line says that the $Group()$ constructor is a category (so a list of signatures is expected) and that this category is built upon the category Set, i.e., it will have all the Set signatures along with those that are going to be specified in the lines that follow.

The second line is a "−−" comment. It is a comment line that is private to the source code and is intended for the implementer.

The third line gives the signature for a group operation. The $ in source code denotes the domain or category under construction.

The fourth line is a "++" comment (also called "documentation"). It

is a line that is public and can be used by various data base and browser facilities when a user asks for information about a given domain or category.

The operation "/" will be given by a "default definition" (see below)

The line following the internal comment "attributes" sets a system-wide attribute for all operations which are implemented as group operations. Attributes are qualities that an object might have which can be tested in future source code which builds upon previous source code. Recall the tests "If R is a Field then" and "else if R is a EuclidianDomain" from (1.1).

Finally, there is a "default definition" given by the last two lines. Any domain which implements the *Group*() signatures will automatically have an implementation of "division" It should be remarked that this discussion has been for the purposes of illustration and while the category *Groups*() actually exists in AXIOM, it has a somewhat different (but similar in spirit) implementation.

```
Group() : Category == Set with
       -- operations
       * : ($, $) -> $
       ++  x * y returns the product

       inv : $ -> $
       ++ inv(x) returns the inverse

       1 : () -> $
       ++ 1 is the identity element

       / : ($, $) -> $
       ++ x/y  = x * inv(y)
       ++ it is defined in all groups by a default definition

       --  attributes
       associative(*)

     add
       x:$ / y:$ == x * inv(y)
```

Given the AXIOM category Group, one does not necessarily have any group! To obtain a group, one must

- i. implement the group signatures or
- ii. call some "group constructor".

The same holds for all constructors. To explain, one might write

$$Q := Fraction(Integer) \tag{4}$$
$$G := Heisenberg(Q) \tag{5}$$

to obtain the group of 3×3 upper triangular matrices with ones along the diagonal and rational number entries. This is an example of how one calls a group constructor (viz., $Heisenberg(R)$) which has already been implemented in the system. In passing, also note that the constructor $Fraction(R)$ where R is an integral domain already exists in AXIOM. It returns a domain which represents the field of fractions of R.

Given a domain such as G above, one works with it as follows. To declare some elements to be of type G write

$$(x, y, z) : G$$

To make these variables the standard group generators write

$$x := generator(1) \tag{6}$$
$$y := generator(2) \tag{7}$$
$$z := generator(3) \tag{8}$$

(this assumes that the signature $generator : NNI \to G$ with the obvious usage exists in the domain G). To create some words in G, for example, write

$$x * inv(y) * x.$$

There is also the notion of a "package" in AXIOM. A package is exactly like a domain in the sense that certain signatures are implemented but it only exports operations that work on objects supplied by other domains (it has no signatures with "$" in them). The purpose of a package is to group together a collection of procedures which may be used for various specialized

tasks. For example, one might have a package to implement isomorphisms (coercions) of several types between one domain and another.

2.2 Cosmetic Issues

There is a very rich domain in AXIOM called *OutputForm*. Its purpose is to provide the implementor with convenient facilities for displaying an object in a desired form. For example, the free abelian group on n generators $\{t_1, \ldots, t_n\}$ written multiplicatively might be implemented by defining operations on lists of integers $[i_1, \ldots, i_n]$. In any output, it is desirable to represent this list of integers as $t_1^{i_1} \ldots t_n^{i_n}$. AXIOM provides for this through the *OutputForm* domain in connection with a *coercion mechanism*.

Every domain must either inherit or implement a signature of the form

```
coerce : $ -> OutputForm.
```

The purpose of this is to let the AXIOM system know how to output an expression from the given domain. For example, in the free abelian group case above, an implementor might require parameters in the free abelian group constructor which the user is to supply and which will be used to determine the generators of the group. The constructor and its parameters might look like `FreeAbelian(var,dim)` where `var` is of type `Symbol` and `dim` is of type +NNI+. A call to this constructor might look like

```
fab4 := FreeAbelian(t,4).
```

In order to accomplish the desired output, the implementor could write something like the following lines.

```
macro\ O ==> OutputForm
listGens := [sub(t::O,k::O) for k in 1..dim]

coerce(x:$):O ==
  x=1 => 1::O
  reduce(*, [listGens.k ** x.k::O for k in 1 .. dim | not zero? x.k]).
```

The first line is a macro. It allows the substitution of `O` for the longer `OutputForm` (macros are also possible in the interpreter).

The second line takes the user input symbol t and coerces it to O and also coerces (coercions can be forced by the "::" construct) the NNI, k to O

and then forms a list of t subscripted by k as k ranges through the values 1 to dim. In other words, it simply forms the list of "output forms" $[t_1, \ldots, t_n]$.

The next few lines indicate that there may be some other bits of code which the implementor may have written and the next line starts the implementation of the coercion routine for the domain which is under construction.

The next line states that if the expression is equal to the identity element of the group, then the expression "1" should be output.

The next line applies the multiplication operator $*$ *from OutputForm* to the list $[t_1^{x.1}, \ldots, t_n^{x.n}]$ which is itself formed using the operation of exponentiation from O.

The multiplication operator from O takes two output forms and returns an output form with just enough space between them to look reasonable for representing the product of two elements. Similarly, the exponential of an output form by another creates an output form which will display in the usual fashion.

As was mentioned earlier, the domain $OutputForm$ is quite rich. It contains operations to put boxes around objects, place spaces horizontally or vertically, overline or underline, and so on. In addition, the AXIOM system provides for TEX output.

§3. Applications to Research in Algebra

A general method which can be used to attack problems in several areas of mathematics is called "perturbation theory". The basic idea is to realize a given problem as a "perturbation" of a problem with a known solution and then perturb the known solution to obtain a solution to the original problem.

Homological perturbation theory can be applied to obtain small resolutions over several classes of algebras. In this context, one has an algebra A' which has the same underlying R-module structure as an algebra A. One also has a module M' over A' with the same underlying R-module structure as a module M over A (examples will be given later). The objective is to "perturb" a free resolution of M over A to a free resolution of M' over A'. Thus, one is given

$$(A \otimes X, d) \longrightarrow M \longrightarrow 0$$

and one seeks

$$(A' \otimes X', d') \longrightarrow M' \longrightarrow 0$$

obtained from the original resolution by some process. A formal solution to this problem exists [**Lambe2**], [**Lambe3**], [**Barnes-Lambe**]. In order to

discuss these concepts further, some notation will be necessary.

3.1 Homological Perturbation Theory

Let (X, d) denote a differential graded module X over some commutative ring R with unit. Thus $X = \{X_n\}_{n\geq 0}$ is a sequence of R-modules and there are R-module maps

$$\ldots X_{n+1} \xrightarrow{d_{n+1}} X_n \xrightarrow{d_n} X_{n-1} \ldots$$

with

$$d_n d_{n+1} = 0.$$

The differential module (X, d) will often be simply denoted by X. When several modules are involved and the differentials need to be distinguished, they will be written as d_X, etc.

Given two differential modules M and X, M is said to be a *strong deformation retraction* (SDR) of X when there exist maps

$$M \xrightarrow{\nabla} X, \quad X \xrightarrow{f} M$$

and

$$X \xrightarrow{\phi} X$$

such that the following identities hold.

$$\nabla f = 1_M, \quad f\nabla = 1_X - (d\phi + \phi d),$$

$$f\phi = 0, \quad \phi\nabla = 0, \quad \phi\phi = 0.$$

A fundamental theorem on transferring differentials from one object to another is the "basic perturbation lemma" (see, for example, [**Shih**], [**Brown**], [**Gugenheim**], [**Lambe-Stasheff**], [**Hübschmann**], [**Barnes-Lambe**]).

3.1.1. Theorem (Basic Perturbation Lemma): *Given SDR–Data*

$$((X, d) \underset{f}{\overset{\nabla}{\rightleftarrows}} (Y, d), \phi)$$

and a new differential D on Y, let $t = D - d$ and more generally

$$t_{n+1} = (t\phi)^n t, \quad n \geq 0.$$

For each n, define (on X)

$$\begin{aligned} \partial_n &= d + f(t_1 + t_2 + \ldots t_{n-1})\nabla \quad (9) \\ \nabla_n &= \nabla + \phi(t_1 + t_2 + \ldots t_{n-1})\nabla \quad (10) \end{aligned}$$

and (on Y)

$$\begin{aligned} f_n &= f + f(t_1 + t_2 + \ldots + t_{n-1})\phi \quad (11) \\ \phi_n &= \phi + \phi(t_1 + t_2 + \ldots t_{n-1})\phi. \quad (12) \end{aligned}$$

If the limit exists, new SDR–data

$$((X, \partial_\infty) \underset{f_\infty}{\overset{\nabla_\infty}{\rightleftarrows}} (Y, d + t), \phi_\infty)).$$

is obtained.

Remark: There are situations in which the limit does not exist, but there are many classes of examples for which it does. One criterion that is often used is the existence of a filtration for which $t\phi$ can be shown to be nilpotent in any finite degree. There are however other criteria ([**Gugenheim**], [**Barnes-Lambe**]).

To apply this algorithm to the case of "perturbing" a given resolution as mentioned above, the following method can be attempted.

(3.1.2) Find an SDR of X into the bar construction (see below) of A

$$(A \otimes X \underset{f}{\overset{\nabla}{\rightleftarrows}} B(A), \phi)$$

(3.1.3) Use the additive isomorphism $B(A) \cong B(A')$ to "transfer" the differential of $B(A')$ down to $A \otimes X'$ using the formulae in (3.1.1).

This method has been discussed in some detail in [**Lambe2**] and [**Lambe3**].

3.2. Application to Group Cohomology

Consider the following class of groups G_q and H from [**Lambe2**]. For a fixed integer q, let

$$G_q = \left\{ \begin{pmatrix} 1 & x & q^{-1}z \\ 0 & 1 & y \\ 0 & 0 & 1 \end{pmatrix} \mid x, y, x \in Z \right\}$$

and give G_q the operation of matrix multiplication. This is an example of a polynomial group law on $\mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} = \mathbb{Z}^3$.

Let

$$H = \mathcal{G} \times \mathbb{Z}$$

where $\mathbb{Z}$ is the additive group of integers and $\mathcal{G} = \mathbb{Z}+i\mathbb{Z}$ is the additive group of Gaussian integers. The multiplication is given as follows. Let $g_1 = x_1+x_2i$, $g_2 = y_1 + y_2i$.

$$(g_1, x_3)(g_2, y_3) = (g_1 + i^{x_3}g_2, x_3 + y_3).$$

Note that

$$i^{x_3}g_2 = y_1s_1 - \frac{x_3y_2\pi}{2}s_2 + (y_2s_1 + \frac{x_3y_1\pi}{2}s_2)i$$

where

$$s_1 = \sum \frac{(-1)^k(x_3^2\pi^2)^k}{4^k(2k)!} \quad s_2 = \sum \frac{(-1)^k(x_3^2\pi^2)^k}{4^k(2k+1)!}.$$

Thus, here is an example of a convergent power series group law on $\mathbb{Z}^3$.

$$(x_1, x_2, x_3)(y_1, y_2, y_3) =$$

$$(x_1 + y_1, x_2 + y_2, x_3 + y_3) + (\frac{-x_3y_2\pi}{2} + (\frac{x_3y_1\pi}{2})i, 0) + O(\geq 3).$$

More generally, think of groups K with underlying set $\mathbb{Z}^n$ with group law either a polynomial or a convergent power series In any case, think of the group ring $\mathbb{Z}(K)$ as a perturbation of the ring of finite Laurent polynomials

$$A = Z[t_n^{-1}, \ldots, t_1^{-1}, t_1, \ldots, t_n]$$

which is itself the group ring of the free abelian group $(\mathbb{Z}^n, +, 0)$. In the case of the family G_q above, since there is clearly an isomorphism with the family of operations ρ_q on $\mathbb{Z}^3$ given by

$$(x_1, x_2, x_3)(y_1, y_2, y_3) = (x_1 + y_1, x_2 + y_2, x_3 + y_3 + qx_1y_2),$$

one sees that

$$\lim_{q\to 0} G_q \cong \mathbb{Z}^3$$

and $\mathbb{Z}(G)$ is actually a deformation of the ring of Laurent polynomials [**Lambe3**].

The homology of a group G is the homology of the complex

$$\mathbb{Z} \otimes_{\mathbb{Z}(G)} B(\mathbb{Z}(G))$$

where it is recalled that the bar construction resolution $B(A)$ of an algebra A is a free A-module and resolution $B(A) \longrightarrow \mathbb{Z} \rightarrow 0$. It is given by

$$B(A) = A \otimes \bar{B}(A), \quad \bar{B}(A) = \sum_{n=0}^{\infty} \bar{B}_n(A)$$

$$\bar{B}_n(A) = \otimes^n \bar{A}$$

where it is assumed that A is augmented with unit

$$0 \rightarrow R \xrightarrow{\sigma} A \xrightarrow{\epsilon} R \rightarrow 0$$

and $\bar{A} = A/\sigma(R)$. The differential in $B(A)$ is given by

$$\partial[b_1|\ldots|b_n] = [b_2|\ldots|b_n] + \sum \pm[b_1|\ldots|b_ib_{i+1}|\ldots|b_n] \tag{13}$$

$$\pm[b_1|\ldots|b_{n-1}]. \tag{14}$$

See [**MacLane**] for details about the bar construction.

To compute the necessary formulae in the case of the perturbations of the abelian group law above, one requires the following objects

- i.) The abelian group $\mathbb{Z}^n$.
- ii.) The group $G = (\mathbb{Z}^n, \rho)$, $\rho(x, y) = x + y + \wp(x, y)$.
- iii.) The group ring A of $\mathbb{Z}^n$ (Laurent polynomials).
- iv.) The group ring of G, a perturbation of the Laurent polynomials.
- v.) The bar constructions $B(A)$ and $B(\mathbb{Z}(G))$.
- vi.) An SDR $(A \otimes E[u_1, \ldots, u_n] \underset{f}{\overset{\nabla}{\rightleftarrows}} B(A), \phi)$ (see [**Lambe2**], [**Lambe3**] and the references given there), and
- vii.) The additive isomorphism $B(\mathbb{Z}(G)) \cong B(A)$.

  These objects and the calculations necessary for (3.1.1) have been implemented and carried out in AXIOM. To describe what it looks like, suppose that there are domain constructors

- `FreeAb(n)`, (the free abelian group of rank n),

- `CEMTKRes(n)`, (Cartan-Eilenberg-MacLane-Tate-Koszul Resolution),
- `BarCons(R, G)`, (the bar construction of `R(G)`),
- `SdrFreeAb(n)`, (the strong deformation retraction mentioned above),

  and
- `LagRes(n, xVar, yVar, p)`, (a package, `LinearAffineGroupResolution`), which implements the additive isomorphism needed in (3.1.3).

The domain constructor `FreeAb(n)` represents the free abelian group on n generators $t_1, \ldots, t_n$ written multiplicatively. It has signatures like

```
generator : NNI -> $
```

where $generator(i) = t_i$, etc. Let

$$A = Z[t_n^{-1}, \ldots, t_1^{-1}, t_1, \ldots, t_n]$$

be the group ring of the free abelian group on n generators. The group ring constructor exists in AXIOM, but so does the monoid ring and, for our purposes, the monoid ring suffices. Thus, take

```
A := MonoidRing(Integer, FreeAb(n)).
```

The domain constructor `CEMTKRes(n)` represents the resolution

$$(A \otimes E[u_1, \ldots, u_n], d, \varphi) \xrightarrow{\epsilon} \mathbb{Z} \to 0$$

where the differential d is given by

$$d(p) = 0, \quad d(u_i) = t_i - 1$$

for $p \in A$ (d is extended as a derivation of the whole algebra).

The augmentation ϵ is given by

$$\epsilon(t^i) = 1, \quad d(t^i u_I) = 0$$

where $u_I = u_{i_1} \ldots u_{i_k}$ for an index set $I = \{i_1, \ldots, i_k\}$ with $k \geq 1$. An explicit contracting homotopy φ exists, and in fact a general formula for it was derived through the use of AXIOM (see [**Lambe3**] and below). This was done

through a sort of "bootstrap" process – recursive formulae were known, and by calculating enough examples a general formula was found. The recursive formulae are rather complex and unwieldy so that hand calculation is overly tedious. In fact, as can be seen from examples below, the closed formulae for φ are also rather complex and tedious to deal with by hand. One may still use a computer to deal with them but, if one has closed formulae (as opposed to recursive formulae), one can write routines that are tremendously more efficient than recursive forms. This is an important issue and we will surely see more and more of this sort of bootstrap process in many areas of application in the future.

The domain `CEMTKRes(n)` has the signatures

```
generator    : NNI -> $
diff         : $  -> $
phi          : $  -> $
terms        : $  -> List($)
homogeneous? : $  -> Boolean
```

among others. They have the obvious meanings. The function $diff$ gives the differential d above, *terms* gives a list of the terms in a given expression, and *homogeneous*? checks to see if a given expression is of one degree (and not a linear combination of terms of mixed degrees).

The domain constructor $BarCons(R, G)$ has the expected signatures, including the standard contracting homotopy s.

The package constructor $SdrFreeAb(n)$ implements the SDR

$$(A \otimes E[u_1, \ldots, u_n] \underset{f}{\overset{\nabla}{\rightleftarrows}} B(A), \phi)$$

mentioned above. It has signatures

```
inclusion  : CEMTKRes(n) -> BarCons(Integer, FreeAb(n))
projection : BarCons(Integer, FreeAb(n)) -> CEMTKRes(n)
homotopy   : BarCons(Integer, FreeAb(n)) -> BarCons(Integer, FreeAb(n))
```

among others.

The package constructor `LagRes(dim,xVar,yVar,p)` implements several auxiliary signatures necessary to carry out the steps of the perturbation method mentioned above. It has

```
barG2barZ    :  barG -> barZ
barZ2barG    :  barZ -> barG
basis        :  () -> List(CEMTKRes(n))
t            :  barZ -> barZ
thpi         :  barZ -> barZ
```

among other signatures, where

```
barZ ==> BarCons(Integer, FreeAb(n))
barG ==> BarCons(Integer, G)
```

and G is the group whose polynomial group law is given by the polynomial p in arguments $xVar$ and $yVar$. The function $barG2barZ$ implements the additive isomorphism mentioned above in (3.2) vii.), and $barZ2barG$ is its inverse. The function *basis* gives the list of basis (over A) elements for the exterior algebra

$$\{1, u_1, \ldots, u_n, u_1u_2, \ldots, u_1u_2 \ldots u_n\}$$

(2^n elements). It takes no arguments. The function t implements the difference of the abelian group law differerential and the "perturbed" group law differential needed for (3.1.1) and the function *thpi* is the composite $t\phi$ mentioned in (3.1.1).

It should be pointed out that these domains and packages as well as some of the categories and domains on which they were built were not a part of the basic AXIOM system distributed by NAG. The author needed to implement all the necessary objects, but it was possible to do this in a very convenient manner following the ideas already outlined as well as those outlined in [**Lambe1**]. The complete source code is available from the author.

Assuming that we have all the above, here is an actual AXIOM input file which does the calculation for the group G of upper triangular 3×3 matrices in (3.2) at the value $q = 1$.

```
-- alternate names for some domains used
L := List
P := Polynomial
I := Integer

-- Set up for the Heisenberg Group
```

```
dim := 3
fab := FreeAb dim
cem := CEMTKRes dim
bar := BarCons(I, fab)
sdr := SdrFreeAb dim

xVar: L Symbol := [x1, x2, x3]
yVar: L Symbol := [y1, y2, y3]

-- Give the group law
p : L P I := [x1 + y1, x2 + y2, x3 + y3 + x1 * y2]

respkg   := LagRes(dim, xVar, yVar, p, inv)
extBasis := basis()$respkg
image    := [inclusion(l)$sdr  for l in extBasis]
timage   := [t(l)$respkg for l in image]

extRank := 2**dim - 1

-- We now need to apply tphi iteratively
-- until each term vanishes, i.e., until
-- (tphi)**n(timage) = 0

lim: L bar := [ [ ] for i in 1..extRank]

for i in 1..extRank repeat
   tmp   := timage.i
   tmp   := tphi(tmp)$respkg
   lim.i := cons(tmp, lim.i)
   while not(tmp = 0) repeat
      tmp   := tphi(tmp)$respkg
      lim.i := cons(tmp,lim.i)

-- Now compute the projections
ftimage := [projection(l)$sdr for l in 1..extRank]

partials: L L cem := [ [ ] for i in 1..extRank]
```

```
for i in 1..extRank repeat
   partials.i := [proj(k)$sdr for k in lim.i]

-- Now the differentials
dcem   := [diff l for l in extBasis]
dinfty := [ dcem.i + ftimage.i + reduce(+, partials.i)
                  for i in 1..extRank]
```

A similar calculation can be set up and run for the power series group law for H above. The results of these calculations are quite orderly. For the power series group law we have obtained the resolution

$$\begin{aligned}
(Z(G) &\otimes E[u_1, u_2, , u_3], d) && (15)\\
d(u_i) &= t_i - 1 && (16)\\
d(u_1u_2) &= (t_1 - 1)u_2 - (t_2 - 1)u_1 && (17)\\
d(u_1u_3) &= (t_2 - 1)u_3 + u_2 + t_3u_1 && (18)\\
d(u_2u_3) &= -(t_1^{-1} - 1)u_3 - t_3u_2 - t_1^{-1}u_1 &&\\
d(u_1u_2u_3) &= u_2u_3 - (t_1^{-1} - 1)u_1u_3 - (t_1 - t_3)u_1u_2. && (19)
\end{aligned}$$

Clearly, the homology (or cohomology), for example, of the group H with $\mathbb{Z}$ coefficients can be easliy read from this.

§4. Abstractly Symbolic Domains of Computation

Given the power and convenience of a system like AXIOM, one is tempted to wonder if the type of calculation done above for an individual group can be achieved for a parameterized class of groups like the class G_q in (3.2) which is parameterized by q. At least a part of the issue in doing that involves some difficulties in manipulating the SDR data from (3.2) vi.) in a way that allows one to keep q "symbolic". To understand this, consider the case $n = 1$ of the SDR in (3.2) vi.).

$$(A \otimes E[u] \underset{f}{\overset{\nabla}{\rightleftarrows}} B(A), \phi)$$

where

$$A = \mathbb{Z}[t^{-1}, t].$$

The maps f and ϕ crucial to the perturbative method outlined above are completely determined by a contracting homotopy φ on $A \otimes E[u]$. In fact, this is true in much more generality (see [**Lambe3**] for details). The equation that determines a contracting homotopy is

$$d\phi + \phi d = 1.$$

In this one-dimensional case, it is not hard to solve this equation for φ given d and the augmentation already mentioned (for $CEMTKRes(n)$ in (3.2)). In fact, a solution is

$$\varphi(t^i) = \frac{t^i - 1}{t - 1}u, \quad \varphi(t^i u) = 0.$$

Now ∇, f, and ϕ are determined by the formulae

$$\nabla(u_I) = s\nabla(du_I), \quad f(\bar{b}) = sf(\partial\bar{b}), \quad \phi(\bar{b}) = -s(\nabla f(\bar{b}) + \phi\partial(\bar{b}))$$

defined on "reduced elements", i.e., elements of $E[u_1, \ldots, u_n]$ and $\bar{B}(A)$ and extended A-linearly over all their domains. In the one-dimensional case, it is easy to see that a formula for f is therefore

$$\begin{aligned} f([t^i]) &= \tfrac{t^i-1}{t-1}u \\ f([t^{i_1}|\ldots|t^{i_k}]) &= 0, \quad \text{if } k > 1. \end{aligned} \tag{20}$$

Now the higher-dimensional cases of this are also needed. For example, the 3-dimensional case is needed for the class G_q, In dealing with the formulae in (3.1.1) some functions need to be iterated over quantities involving the coefficient $\frac{t^i-1}{t-1}$. Of course, to evaluate these iterations, one can expand

$$\frac{t^i - 1}{t - 1} = \sum_{j=0}^{i-1} t^j,$$

but then one has to wonder about representing not only the symbolic exponent, but now also the symbolic limit on the summation! This is a serious issue, particularly because in the iterations necessary one does indeed encounter applications of φ to coefficients like $\frac{t^i-1}{t-1}$. One solution that has been quite successful is to encode the "symbolic summation" as

$$t^{\{i\}} = \frac{t^i - 1}{t - 1}$$

for i any integer. Thus,

$$f(t^i) = t^{\{i\}_1}u$$

and *by definition* we take

$$f(t^{\{i\}_1}) = t^{\{i\}_2}u$$

and so on to recursively define $t^{\{i\}_n}$. For the purposes of the perturbation calculations necessary in this class of problems, there is an "algebra of exponents" for such "symbolically exponentiated" quantities and, given the flexible extensibility of the AXIOM system, it has been possible to develop an actual domain of computation for such symbolic calculations. With these "symbolic domains", it has been possible to calculate closed formulae for the perturbation steps needed for the resolution calculations mentioned above. For example, a class of resolutions parameterized by q for the class of groups G_q parameterized by q may be derived. These 3-dimensional results are summarized below. See [**Lambe2**] and [**Lambe3**] for more complete results and details.

The SDR

$$(A \otimes E[u_1, u_2, u_3] \underset{f}{\overset{\nabla}{\rightleftarrows}} B(A), \phi)$$

for the group ring

$$A = Z[t_3^{-1}, t_2^{-1}, t_1^{-1}, t_1, t_2, t_3]$$

is given as follows. The differential d is

$$\begin{aligned} d(p) &= 0 && (21)\\ d(u_i) &= t_i - 1. && (22) \end{aligned}$$

The contracting homotopy φ on $A \otimes E[u_1, u_2, u_3]$ is given by

$$\begin{aligned} \varphi(t^{i_1}t^{i_2}t^{i_3}) &= t^{\{i_1\}}u_1 + t_1^{i_1}t_2^{\{i_2\}}u_2 + t_1^{i_1}t_2^{i_2}t_3^{\{i_3\}}u_3 && (23)\\ \varphi(t^{i_1}t^{i_2}t^{i_3}u_1) &= t_1^{i_1}t_2^{\{i_2\}}u_2u_1 + t_1^{i_1}t_2^{i_2}t_3^{\{i_3\}}u_3u_1 && (24)\\ \varphi(t^{i_1}t^{i_2}t^{i_3}u_2) &= t_1^{i_1}t_2^{i_2}t_3^{\{i_3\}}u_3u_2 && (25)\\ \varphi(t^{i_1}t^{i_2}t^{i_3}u_3) &= 0 && (26)\\ \varphi(t^{i_1}t^{i_2}t^{i_3}u_1u_2) &= t_1^{i_1}t_2^{i_2}t_3^{\{i_3\}}u_1u_2u_3 && (27)\\ \varphi(t^{i_1}t^{i_2}t^{i_3}u_1u_3) &= 0 && (28)\\ \varphi(t^{i_1}t^{i_2}t^{i_3}u_2u_3) &= 0 && (29)\\ \varphi(t^{i_1}t^{i_2}t^{i_3}u_1u_2u_3) &= 0. && (30) \end{aligned}$$

The inclusion, (related to the well-known "shuffle product"), is

$$\begin{aligned}
\nabla(u_i) &= [t_i] && (31)\\
\nabla(u_i u_j) &= [t_i|t_j] - [t_j|t_i] && (32)\\
\nabla(u_1 u_2 u_3) &= [t_1|t_2|t_3] - [t_1|t_3|t_2] - [t_2|t_1|t_3] && (33)\\
&+ [t_2|t_3|t_1] + [t_3|t_1|t_2] - [t_3|t_2|t_1]. && (34)
\end{aligned}$$

The projection in one degree is

$$f([t^{i_1}t^{i_2}t^{i_3}]) = t_1^{i_1}t_2^{i_2}t_3^{\{i_3\}}u_3 + t_1^{i_1}t_2^{\{i_2\}}u_2 + t_1^{\{i_1\}}u_1.$$

The other degrees for the projection as well as a formula for the homotopy will be omitted here since they are given in the references already mentioned. In this dimension it is not overly difficult, but still tedious, to compute these formulae by hand. This case and higher dimensions were successfully (and conveniently) computed entirely in AXIOM.

Consider the class of groups

$$G_q = \left\{ \begin{pmatrix} 1 & x & q^{-1}z \\ 0 & 1 & y \\ 0 & 0 & 1 \end{pmatrix} \mid x, y, z \in \mathbb{Z} \right\}.$$

once again. A resolution of $\mathbb{Z}$ over $\mathbb{Z}(G_q)$ is given by the complex (computed entirely in AXIOM)

$$(\mathbb{Z}(G_q) \otimes E[u_1, u_2, u_3], d)$$

where

$$\begin{aligned}
d(u_i) &= t_i - 1 && (35)\\
d(u_1u_2) &= (-t_1t_2t_3^{\{q\}})u_3 + (t_1 - 1)u_2 - (t_2 - 1)u_1 && (36)\\
d(u_1u_3) &= (t_1 - 1)u_3 - (t_3 - 1)u_1 && (37)\\
d(u_2u_3) &= t_2 - 1)u_3 - (t_3 - 1)u_2 && (38)\\
d(u_1u_2u_3) &= (t_1 - 1)u_2u_3 - (t_2 - 1)u_1u_3 + (t_3 - 1)u_1u_2. && (39)
\end{aligned}$$

From this it is easy, for example, to see that the second homology group of G_q with coefficients in the integers is

$$H^2(G_q; \mathbb{Z}) \cong \mathbb{Z} \oplus \mathbb{Z} \oplus \mathbb{Z}/q$$

for any q.

This calculation was achieved through the use of several symbolic domains of computation. First, there is `SymbSdrFreeAb` with signatures

```
pow  : ($, Expression Integer) -> $
sExp : (Expression Integer, NonNegativeInteger) -> $
```

where, given any symbolic group element $g = t_1^{i_1} \dots t_n^{i_n}$ and any expression e (with integer coefficients), the operation $pow(g, e)$ returns $t_1^{i_1 e} \dots t_n^{i_n e}$ and given an expression e, and a non-negative integer i such that $1 \leq i \leq n$, $sExp(e, i) = t_i^{\{e\}}$ (iterated brackets $t_i^{\{e\}_k}$ are handled by simply iterating brackets – they are counted up on output and a sensible output form is chosen). Of course, this domain also has all the usual operations as well. Next, there is $SymbCEMTKRes$. This domain has all the same signatures as its counterpart $CEMTKRes$, except things have been rearranged internally so that if one calls the contracting homotopy *phi* on a "symbolic group ring element" t_j^i, then the correct *symbolic expression* involving $t_j^{\{i\}}$ is returned. There are also $SymbBarCons$, and $SymbSdrFreeAb$. Finally, there is the symbolic domain constructor

$$SymbGroup(dim, groupLaw)$$

where

```
&dim : NonNegativeInteger \cr
&groupLaw : (LEI, LEI) \to LEI
```

and `LEI:= List\ Expression\ Integer`. It contains signatures

```
symbDiff : SBAR -> SBAR
t : SBAR -> SBAR
```

among others, where $SBAR$ is the symbolic bar construction domain for this (symbolically represented) group. The coefficient ring for $SBAR$ is $ExpressionInteger$. As before, t is the function which gives the difference between the bar construction differential for this group and the (symbolic) free abelian group.

With these domains in place, the above output for the class G_q is quite easily computed using essentially the same input as in (3.2). One needs to change the first few lines to

---

```
-- Set up for general groups of rank 3
--
```

```
dim := 3
fab := SymbFreeAb dim
cem := SymbCEMTKRes dim
bar := SymbBarCons(Expression Integer, fab)
sdr := SymbSdrFreeAb dim
\verbfile{symbsdr}
```

Finally, it should be pointed out that the perturbation method does not just give rise to resolutions that can be used to compute Ext and Tor abstractly, it actually gives a strong comparison with the bar construction, i.e., this comparison is actually a part of what gets computed. In the case of the types of group rings presented here (polynomial and convergent power series group laws), it actually *computes* an SDR of the new small resolution in the bar construction. From this, one may conveniently read off bar construction cycles that represent homology classes of the group. In cases where the group is actually a $K(\pi, 1)$-manifold, as is the case for the classes mentioned here, this calculation then actually leads to an explicit cell structure for the manifold (by realizing the simplicial version of the bar construction). We will not go into details here, but will simply end with an example that arose in some conversations with Ronnie Brown and Graham Ellis at the University of Wales, Bangor, UK recently. It concerns the group $F_{3,2}$ which is the free group F_3 on three generators modulo the third term of its lower central series (thus $F_{3,2}$ is the free-nilpotent 3-generated 2-step nilpotent group).

First the complex (just in degrees 3 and 4) for computing the homology is given. It was derived from a small resolution of the integers over the group ring (obtained by homological perturbation theory). In fact, the complex is so small and sparse, that one can compute the homology easily by sight. Here is the complex:

$$\begin{aligned}
\overline{d}\,(u_1\ u_2\ u_3) &= u_5\ u_6 + u_4\ u_6 + u_4\ u_5 + u_3\ u_4 - u_2\ u_5 + u_1\ u_6 && (40)\\
\overline{d}\,(u_1\ u_2\ u_5) &= -u_4\ u_5 && (41)\\
\overline{d}\,(u_1\ u_2\ u_6) &= -u_4\ u_6 && (42)\\
\overline{d}\,(u_1\ u_3\ u_4) &= +u_4\ u_5 && (43)\\
\overline{d}\,(u_1\ u_3\ u_6) &= -u_5\ u_6 && (44)\\
\overline{d}\,(u_2\ u_3\ u_4) &= +u_4\ u_6 && (45)\\
\overline{d}\,(u_2\ u_3\ u_5) &= +u_5\ u_6 && (46)
\end{aligned}$$

$$
\begin{array}{lll}
\overline{d}\,(u_1\ u_2\ u_3\ u_4) & = & +u_4\ u_5\ u_6 + u_2\ u_4\ u_5 - u_1\ u_4\ u_6 \\
\overline{d}\,(u_1\ u_2\ u_3\ u_5) & = & -u_4\ u_5\ u_6 + u_3\ u_4\ u_5 - u_1\ u_5\ u_6 \\
\overline{d}\,(u_1\ u_2\ u_3\ u_6) & = & +u_4\ u_5\ u_6 + u_3\ u_4\ u_6 - u_2\ u_5\ u_6 \\
\overline{d}\,(u_1\ u_2\ u_5\ u_6) & = & -u_4\ u_5\ u_6 \\
\overline{d}\,(u_1\ u_3\ u_4\ u_6) & = +u_4\ u_5\ u_6 & \\
\overline{d}\,(u_2\ u_3\ u_4\ u_5) & = & -u_4\ u_5\ u_6
\end{array}
\tag{47}
$$

Clearly, the homology in dimension three has generators

$$u_1u_2u_4, u_1u_2u_5 + u_1u_3u_4, u_1u_2u_6 + u_2u_3u_4, u_1u_3u_5,$$

$$u_1u_3u_6 + u_2u_3u_5, u_1u_4u_5, u_1u_4u_6, u_1u_5u_6, u_2u_3u_6, u_2u_4u_6, u_2u_5u_6, u_3u_5u_6$$

Thus the rank of H^3 is 12. Cycle representatives in the bar construction for all the generators of all the homology have been computed in AXIOM using homological perturbation. They are obtained by computing the image of the map ∇_∞ from (3.1.1). Only the image of the cycle $u_1u_2u_5 + u_1u_3u_4$ will be given here. Complete results for several other classes of groups and algebras have also been obtained.

$$
\begin{array}{c}
\nabla_\infty(u_1u_2u_5 + u_1u_3u_4) = \\
\left[t_1{}^1|t_2{}^1|t_5{}^1\right] - \left[t_1{}^1|t_5{}^1|t_2{}^1\right] - \left[t_2{}^1|t_1{}^1|t_5{}^1\right] + \left[t_2{}^1|t_5{}^1|t_1{}^1\right] + \left[t_5{}^1|t_1{}^1|t_2{}^1\right] - \left[t_5{}^1|t_2{}^1|t_1{}^1\right] \\
+ \\
-\left[t_1{}^1\ t_2{}^1|t_4{}^1|t_5{}^1\right] + \left[t_1{}^1\ t_2{}^1|t_5{}^1|t_4{}^1\right] - \left[t_5{}^1|t_1{}^1\ t_2{}^1|t_4{}^1\right] \\
+ \\
\left[t_1{}^1|t_3{}^1|t_4{}^1\right] - \left[t_1{}^1|t_4{}^1|t_3{}^1\right] - \left[t_3{}^1|t_1{}^1|t_4{}^1\right] + \left[t_3{}^1|t_4{}^1|t_1{}^1\right] + \left[t_4{}^1|t_1{}^1|t_3{}^1\right] - \left[t_4{}^1|t_3{}^1|t_1{}^1\right] \\
+ \\
\left[t_1{}^1\ t_3{}^1|t_4{}^1|t_5{}^1\right] - \left[t_1{}^1\ t_3{}^1|t_5{}^1|t_4{}^1\right] - \left[t_4{}^1|t_1{}^1\ t_3{}^1|t_5{}^1\right]
\end{array}
$$

Current address:
DIMACS, Rutgers University
CoRE Bldg., Frelinghuysen Rd.
Piscataway, NJ 08855-1179
llambe@cesl.rutgers.edu

References

[**Barnes-Lambe**] Barnes, D., and Lambe, L., *Fixed point approach to homological perturbation theory*, Proc. Amer. Math. Soc., 112(1991), 881-892.

[**Brown**] Brown, R., *The twisted Eilenberg-Zilber theorem*, Celebrazioni Archimedee del secolo XX, Simposio di topologia 34-37(1967).

[**Gugenheim**] Gugenheim, V.K.A.M., *On the chain complex of a fibration*, IL. J. Math. 3(1972), 398-414.

[**Hübschmann**] Hübschmann, J., *The homotopy type of $F\Psi^q$, the complex and symplectic cases*, Cont. Math. 55(1986), 487-518.

[**Jenks**] Jenks, R., *MODLISP: An Introduction*, Springer Lecture Notes in Comp. Sci., vol. 72, 466-480.

[**Jenks-Sutor**] Jenks, R., and Sutor, R., **AXIOM**, Springer Verlag, NY, 1992.

[**Lambe1**] Lambe, L., *Scratchpad II as a tool for mathematical research*, Jon Barwise's column in Notices of the Amer. Math. Soc., February, 1989.

[**Lambe2**] Lambe, L., *Homological perturbation theory, Hochschild homology and formal groups*, Proc. Conference on Deformation Theory and Quantization with Applications to Physics Amherst, MA June 1990, Cont. Math., AMS (to appear).

[**Lambe3**], Lambe, L., *Resolutions that split off of the bar construction*, J. Pure & Appl. Alg., (to appear).

[**Lambe-Stasheff**] Lambe, L., and Stasheff, J., *Applications of perturbation theory to iterated fibrations*, Manuscripta Math., 58(1987), 363-376.

[**MacLane**] MacLane, S., **Homology**, Die Grundlehren der Math. Wissenschaften, Band 114, Springer Verlag, NY, 1967.

[**Shih**] Shih, W., *Homology des espaces fibrés*, Inst. des Hautes Études Sci., vol. 13 (1962), pp. 93-176.