Introduction to Magma

Wieb Bosma and John Cannon

Radboud University Nijmegen and University of Sydney

School ECC2017, Nijmegen, November 2017

PART 1

Foundations

Foundations

Magma is designed to support computation in branches of mathematics that borrow heavily from algebra.
Some initial areas of application include:-

- Algebra
- Number theory
- Algebraic geometry
- Lie theory
- Algebraic combinatorics
- Algebraic topology
- Application areas that make heavy use of algebra: e.g., applied graph theory, coding theory, cryptography.

Foundations

The first step was to design and implement a framework for efficiently computing with the structures of modern algebra.

Examples: Groups, rings, fields, modules, ...

This foundation would then provide a basis for computing with a wider class of structures that are defined largely in terms of algebraic structures.

Examples: Geometric varieties, incidence structures, ...

Foundations

We need:

- A type system
- A domain-specific programming language.
- Algorithms for each class of structures installed.

The type system is based on the key objects of modern algebra: algebraic structures and morphisms.

Ideas from universal algebra provide a framework.

Basic Requirements

- 'Mathematical' aggregate data structures, such as magmas, sets, sequences and mappings, should be used.
- The specification of mathematical objects and their attributes are to be *precise* and *unambiguous*.
- The semantics associated with each object definable in the language is to be as close as possible to the standard mathematical interpretation.
- The user language should support common *mathematical notation* as far as possible.
- *Efficiency* is to be a paramount concern.

Characteristics of the Magma Language

The Magma language is an imperative programming language having standard imperative style statements and procedures.

The language has a functional subset providing functions as first class objects, higher order functions, partial evaluation, etc.

Constructors are used to specify magmas, elements of magmas, mappings and aggregates (sets, sequences, records etc.).

Much of the power of the language derives from its *constructors*.

PART 2

Magmas and their Construction

Magmas

From the mathematical viewpoint we have a hierarchy of objects:

1. A class of abstract structures V defined by a common set of axioms. This is called a variety. Eg a module.
2. A concrete realisation of some subclass of A. We call this a category and denote it by C . Eg the class of all finite dimensional vector spaces.
3. A particular structure A of C which we call a magma. Eg a particular finite dimensional vector space.
4. In general the definition of a magma A involves one or more carrier sets. In the case of a vector space there is one set, the set of elements (or vectors).

Magmas

In general a magma can be constructed from some *free* or *universal* magma by means of a chain of submagma, quotient magma and extension operations.

Elements of the carrier set(s) of a magma M have M as their parent.

Each individual magma created during a run defines a separate type in the programming language sense.

Construction of Magmas

Free or *universal* magmas:-

- `VectorSpace(`*field*`,` *degree*`)`
- `PolynomialRing(`*ring*`,` *num indets*`)`
- `MatrixAlgebra(`*ring*`,` *degree*`)`

Derived magmas:-

- `VectorSpace<`*field*`,` *degree* `|` *generators*`>`
- `MatrixAlgebra<`*ring*`,` *degree* `|` *generators*`>`
- `Graph<` *num vertices* `|` *edges* `>`

Magmas created by intrinsics:-

- `FiniteField(`*cardinality*`)`
- `NumberField(`*field*`,` *polynomial*`)`
- `EllipticCurve(`*curve*`,` *point*`)`

Construction of Magmas (cont)

Forming submagmas, quotients or extensions of magmas:

- `sub<` *magma M* | *submagma of M*>
- `quo<` *magma M* | *submagma of M*>
- `ext<` *magma M* | *extension data*>

`quo< M | ... >` returns the quotient magma Q of M and the natural epimorphism.

`sub< M | ... >` returns the submagma and the inclusion monomorphism.

Example: Tower of Finite Fields

```
> K<u> := GF(3); K;
Finite field of size 3

> R<x> := PolynomialRing(K);
> L<v> := ext< K | x^3 + 2*x + 1 >; L;
Finite field of size 3^3

> S<y> := PolynomialRing(L);
> M<w> := ext< L | y^2 + v*y - 1 >; M;
Finite field of size 3^6

> u + v^8 + v^19*w + v^12;
v^19*w + 2
```

Element Constructors

- `elt<` *magma* `|` $expr_1$`, ..., `$expr_k$ `>`
- *magma* `! [` $expr_1$`, ..., `$expr_k$ `]`

Example: Vector space

```
> V := VectorSpace(Rationals(), 3);
> v1 := V ! [ 1, 1/2, 1/3 ]; v1;
(  1 1/2 1/3)
> Parent(v1);
Full Vector space of degree 3 over Rational Field
> v2 := V ! [ 4/5, 2, 3 ];
> v1 + v2;
( 9/5  5/2 10/3)
U := sub< V | v1, v2, 3*v1 - v2 >;
Dimension(U);
2
```

Element Constructors (cont)

Example: Treat the finite field $G = \mathbb{F}_{3^{12}}$ as a 4-dim vector space V over $F = \mathbb{F}_{3^3}$.

```
> G<g> := FiniteField(3, 12);
> F<f> := FiniteField(3, 3);
> V, v := VectorSpace(G, F);
> V, v;
Full Vector space of degree 4 over GF(3^3)
Mapping from: FldFin: G to ModTupFld: V
```

Note that the map v from G to V, is also returned,

```
> e := V ! [ f, f^11, f^2+f+1, 7 ];  e;
(   f f^11  f^6    1)
> e @@ v;
g^86726
> f + f^11*g + (f^2+f+1)*g^2 + 7*g^3;
g^86726
```

PART 3

Aggregate Datatypes

Homogeneous Aggregate Constructors

- { } *Set:* homogeneous, finite, unordered, no duplicates, fast $\in$ test
- {* *} *Multiset:* homogeneous, finite, unordered, fast $\in$ test
- {@ @} *Indexed set:* homogeneous, finite, ordered, no duplicates, fast $\in$ test
- {! !} *Formal set:* homogeneous, unordered, no duplicates, $\in$ test via predicates
- [ ] *Sequence:* homogeneous, finite, ordered, fast indexed access

Heterogeneous Aggregate Constructors

- `[* *]` *List:* inhomogeneous, finite, ordered
- `< >` *Tuple:* inhomogeneous, finite, ordered
- `rec< >` *Record:* inhomogeneous, finite
- — *Associative arrary:* inverted table

Set Constructor

The general set constructor has the form:

{ U | e : x in E | P }

- *U* is the common parent for all elements
- *e* is an expression involving the (local) parameter *x* which ranges over the magma (or set or sequence) *E*
- *E* is a set which defines the desired set
- *P* is a predicate usually involving *x* that filters the elements being selected

Sequence Constructor: Example

```
N := [ ];
n := 2;
while n lt 1000 do
  n +:= 1;
  if &+Divisors(n-1) eq &+Divisors(n+1) then
    Append(~N, [n, #Factorization(n)]);
  end if;
end while;
```

```
N := [ [n, #Factorization(n)]: n in [2..1000] |
     &+Divisors(n-1) eq &+Divisors(n+1) ];
```

Set Constructor: Example

```
> T := {<x,y,z>: x,y,z in [1..100] |
>                  x^2 + y^2 eq z^2};
```

It's really only necessary to let `x` and `y` iterate over [1..100].

```
> T := {<x,y,Isqrt(x^2 + y^2)>:
>               x,y in [1..100] |
>               IsSquare(x^2 + y^2)};
```

It's annoying to have to type $x^2 + y^2$ twice, so we introduce the `where` expression:

```
> T := {<x,y,Isqrt(w)>: x,y in [1..100] |
>       IsSquare(w) where w is x^2 + y^2};
```

Set Constructor Example (cont)

Still too many uninteresting solutions: try harder.

```
> T := [<x,y,Isqrt(w)>:
>    y in [x..100], x in [1..100] |
>    GCD(x, y) eq 1 and IsSquare(w)
>    where w is x^2 + y^2];
> T;

[ <3, 4, 5>, <5, 12, 13>, <7, 24, 25>,
  <8, 15, 17>, <9, 40, 41>, <11, 60, 61>,
 <12, 35, 37>, <13, 84, 85>, <16, 63, 65>,
 <20, 21, 29>, <20, 99, 101>, <28, 45, 53>,
 <33, 56, 65>, <36, 77, 85>, <39, 80, 89>,
 <48, 55, 73>, <60, 91, 109>, <65, 72, 97> ]
```

Mappings

A *mapping* $f : A \to B$:

$$\texttt{map<}\ \mathit{expr}_1\ \texttt{->}\ \mathit{expr}_2\ \texttt{|}\ \mathit{graph}\ \texttt{>}$$

A *partial mapping* $f : A \to B$:

$$\texttt{pmap<}\ \mathit{expr}_1\ \texttt{->}\ \mathit{expr}_2\ \texttt{|}\ \mathit{graph}\ \texttt{>}$$

A *homomorphism* $f : A \to B$, where A and B are magmas:

$$\texttt{hom<}\ \mathit{expr}_1\ \texttt{->}\ \mathit{expr}_2\ \texttt{|}\ \mathit{graph}\ \texttt{>}$$

An *isomorphism* $f : A \to B$, where A and B are magmas:

$$\texttt{iso<}\ \mathit{expr}_1\ \texttt{->}\ \mathit{expr}_2\ \texttt{|}\ \mathit{graph}\ \texttt{>}$$

PART 4

Standard Magmas & their Algorithms

Standard Magmas & their Algorithms

Installing a new family F of objects (magmas) involves:-

- Deciding on a suitable computational representation for members of F.
- Implementing constructions for F.
- Identifying a basic set of operations which are likely to be needed when computing with members of F.
- Developing and implementing efficient algorithms for performing operations.

The vast majority of our effort goes into developing or improving algorithms.

Structures: Rings and Fields

- *Commutative rings*: Polynomial rings, graded rings, ideals, affine algebras, Galois rings.
- *Fields*: Q, number fields, function fields, class fields, Galois fields, and algebraically closed fields.
- *Local fields: p*-adic and local fields; power series, Laurent series rings and Puiseux series; valuation rings; R and C.
- *Modules*: Vector spaces, R-modules (R a field, Euclidean ring, order of a field), *Hom*-modules, homological algebra, tensors and multilinear functions.
- *Lattices*: Integral lattices, lattices over number fields, Lorentz lattices, quadratic forms, binary quadratic forms.

Structures: Groups and Algebras

- *Groups*: Permutation groups, matrix groups, finitely-presented groups, abelian groups, soluble groups, (infinite) polycyclic groups, automatic groups, braid groups.
- *Lie theory*: Root systems, Coxeter groups, reflection groups, Lie groups, finite groups of Lie type, Lie algebras, algebras, quantum groups.
- *Associative Algebras*: General fd associative algebras, finitely presented algebras, matrix algebras, group algebras, basic algebras, quaternion algebras, Clifford algebras.
- *Non-Associative Algebras*: General algebras, Lie algebras, composition algebras, octonian algebras, Jordan algebras.
- *Representation theory*: A-modules (A an algebra), KG-modules (G a finite group); representations of finite groups, Lie groups and Lie algebras;

Structures: Geometry and Number Theory

- *Geometry*: Convex polytopes and polyhedra, incidence and coset geometries, finite planes.
- *Algebraic geometry*: Schemes, coherent sheaves, algebraic curves, algebraic surfaces, toric varieties.
- *Arithmetic geometry*: Conics, genus 1 curves, elliptic curves, hyperelliptic curves and their Jacobians, some families of surfaces.
- *Modular forms*: Modular curves, modular forms, Hecke modules, Brandt modules, modular abelian varieties, Hilbert modular forms, modular forms over imaginery quadratic fields, congruence subgroups of $PSL(2, R)$.
- *L-functions and Galois representations*: Dirichlet and Hecke characters, L-functions, hypergeometric motives, Artin representations, local Galois representations, admissible representations of $GL_2(Q_p)$.

Structures: Codes, Combinatorics

and Numerical Computing

- *Combinatorics*: Enumerative combinatorics, partitions and Young tableaux, symmetric functions.
- *Incidence structures*: Graphs, multigraphs, networks, incidence structures, designs, finite planes, Hadamard matrices.
- *Coding theory*: Codes over finite fields, codes over finite rings, AG-codes, LDPC codes, additive codes, quantum codes.
- *Arbitrary precision numerical evaluation of standard functions*: Trigonometric and transcendental functions; elliptic and modular functions; theta functions, etc.
- *Arbitrary precision numerical linear algebra*: RQ- and QL-decomposition; rank, kernel, inverse, pseudoinverse of a matrix; eigenvalues and singular value decomposition of a matrix; solution of systems of linear equations.

Magma Information

- *Caching Attributes:* When a user invokes an intrinsic that computes a property of a magma, it will normally store the result as part of the magma's record. This avoids possibly expensive recomputation upon subsequent requests.
- *Asserting Attributes:* In many cases the user can assert such an attribute in advance and so avoid computing it anew on every run. This can save a great deal of time in subsequent runs.
- *Magma Relationships:* It is frequently the case that, given a magma M, other magmas related to M are constructed. To keep track of these relationships a global table is maintained.

Mathematical Databases

- Magma includes about 50 databases (tables) of mathematical objects.
- Typically, such a database contains a complete classification of all structures of some given type up to a specified bound.
- A number of these databases are an integral part of algorithms installed in Magma. For example, factorisation of the integers $2^n \pm 1$ can be sped up enormously if information for that value of n is stored in the tables.

A few of the databases are listed below.

Geometry and Topology

- **Cremona database of Elliptic Curves**: All isogeny classes of elliptic curves having conductor up to 400,000.
- **Stein-Watkins Database of Elliptic Curves:** The Stein-Watkins database of 136,924,520 elliptic curves of conductor up to 10^8.
- **K3 Surfaces:** This comprises a collection of 24,099 K3 surfaces.
- 3**-folds:** Machinery allows the user to generate lists of Fano 3-folds and Calabi–Yau 3-folds.
- **Fundamental Groups of 3-manifolds:** The 11,126 small-volume closed hyperbolic manifolds of the Hodgson-Weeks census.

Number Theory

- **Cunningham Factorizations**: Contains 237,578 factors f of numbers $a^n \pm 1$, where $a < 10000$, $n < 10000$, and $f > 10^9$.
- **Irreducible polynomials**: Sparse irreducible polynomials over $GF(2)$ for all degrees up to 23,030.
- **Conway polynomials**: Conway polynomials for F_2 to $F_1 27$.
- **Galois Polynomials**: For each transitive group G with degree between 2 and 15, the database contains a univariate polynomial over the integers which has G as its Galois group.
- **Sloane-Nebe Lattices:** The lattices of Sloane and Nebe, containing the automorphism group and Θ-series for many examples.

Group Theory

- **Small Groups**: All groups of order up to 2000, except order 1024
- **The ATLAS Database:** Representations of nearly simple groups.
- **Almost Simple Groups:** Almost simple groups stored with their automorphism groups and maximal subgroups.
- **Perfect Groups** : Perfect groups of order up to a million.
- **Transitive Groups**: Transitive permutation groups of degree up to 47.
- **Primitive Groups**: Primitive groups of degree up to 4000.

Group Theory (cont)

- **Irreducible Matrix Groups**: Irreducible subgroups of $GL(n, p)$ where p is prime and $p^n \leq 2499$.
- **Irreducible Soluble Groups**: The irreducible soluble subgroups of GL(n, p) for $n > 1$ and $p^n < 256$.
- **Finite Groups of Rational Matrices:** The maximal finite subgroups of $GL(n, Q)$, for n up to 31.
- **Quaternionic Matrix Groups:** The finite absolutely irreducible subgroups of $GL_n(\mathcal{D})$ where $\mathcal{D}$ is a definite quaternion algebra whose centre has degree d over Q.

Incidence Structures and Codes

- **Simple Graphs:** An interface to the graph enumeration code of Brendan McKay which allows the user to rapidly construct all simple graphs on a given number of vertices.
- **Strongly Regular Graphs:** A database containing a list of strongly regular graphs.
- **Hadamard Matrices:** All inequivalent Hadamard matrices of degree at most 28, and examples of matrices of all degrees up to 256.
- **Skew Hadamard Matrices:** Known skew Hadamard matrices of degree up to 256.
- **Best Known Binary Linear Codes:** The best known linear codes over F_2 of length up to 256.
- **Best Known Ternary Linear Codes:** The best known linear codes over F_3 and F_4 of length up to 100.

PART 5

Elliptic Curves

Elliptic Curves

Magma has support for elliptic curves over a wide variety of structures. Four Handbook sections (Arithmetic Geometry):

- Elliptic Curves,
- Elliptic Curves over Finite Fields
- Elliptic Curves over $\mathbb{Q}$ and Number Fields
- Elliptic Curves over Function Fields

We will have a closer look at these next.

Documentation and other Sources

- W. Bosma, J. Cannon, C. Fieker, and A.Steel: *The Magma Handbook*, Thirteen Volumes, 5800 pages, Version 2.22, May 2016.
- J. Cannon and C. Playoust: *First Steps in Magma*, 16 pages
- G. Bailey: *Appendix: The Magma Language*, In DMWM, pp 331–356.
- W. Bosma, J. Cannon (Editors): *Solving Problems with Magma*, 220 pages.
- W. Bosma, J. Cannon (Editors): Discovering Mathematics with Magma (DMWM), Springer, 2006.

All to be found at:

`http://magma.maths.usyd.edu.au/magma/documentation/`