Summary of New Features in Magma V2.9

May 3, 2002

1 Introduction

This document provides a terse summary of the new features installed in Magma for release version V2.9 (May 3, 2002). Previous releases of Magma were: V2.8 (July 31, 2001), V2.7 (June 30, 2000), V2.6 (November 8, 1999), V2.5 (July 7, 1999), V2.4 (December 3, 1998), V2.3 (January 30, 1998), V2.20 (April 18, 1997), V2.10 (October 14, 1996), V2.01 (June 21, 1996), and V1.30 (March 5, 1996).

2 Summary

Groups

- *Permutation Groups:* A large number of optimizations and improved algorithms have been implemented. For example, the performance of the Schreier-Todd-Coxeter-Sims algorithm for computing the order of a permutation group has been greatly improved in a number of important contexts. A new algorithm has been developed for chief series by Bill Unger which provides greatly improved performance in many situations.

- *Matrix Groups:* A number of new functions are provided for matrix groups. These include functions for computing chief series and chief factors, soluble radical and radical quotient. Again the performance of the Schreier-Todd-Coxeter-Sims algorithm has been upgraded.

- *Finitely Presented Groups:* The handling of subgroup relationships has been revised completely, yielding improved performance and greatly increasing the applicability of key many functions for fp-groups. A fast algorithm for computing the normal subgroups of a finitely presented group G that correspond to submodules of an $\mathbf{F}_p[G]$-module has been installed.

- *Polycyclic Groups:* The collection algorithm for polycyclic groups was improved. Magma now uses a hybrid algorithm written by Volker Gebhardt, which increases the speed of most computations in polycyclic groups by a factor of 5 - 10.

- *Braid Groups:* A new category has been created for braid groups. This new category is intended mainly for cryptographic applications. Emphasis was put on a fast implementation of the group operations and other functions needed for public key cryptosystems based on braid groups.

- *Databases:* A new version of the Small Groups Library written by Besche, Eick and O'Brien has been installed. This database now contains all groups of order up to 2 000, excluding the groups of order 1 024, and several infinite series of larger groups. The transitive groups database now extends up to degree 30, incorporating Alexander Hulpke's latest results.

Basic Rings

- *Finite Fields:* Discrete logarithms in characteristic 2 are now computed using Emmanuel Thomé's record-breaking implementation of Coppersmith's index-calculus algorithm.

Extensions of Rings

- *Number Fields:* Places and divisors for number fields are now implemented.
- *Cyclotomic Fields:* A new representation that permits computation in cyclotomic fields of very large degree is now available.
- *Class Field Theory:* The Magma interface to class field theory (abelian extensions of algebraic number fields) now contains several more functions that exploit the Galois-module structure of the ray-class-groups.
- *Function Fields:* Machinery for working with relative extensions of function fields is a major new feature of this Magma release.

Linear Algebra and Module Theory

- *Sparse Matrices:* A new type is provided for working with sparse matrices.
- *Modules over Orders:* This machinery has been extended to support modules defined over orders of function fields.

Algebras

- A package has been implemented by Willem de Graaf (Utrecht) for computing representations of semisimple Lie algebras.

Algebraic Geometry

- *The Module of Supersingular Points:* A module for computing with the Hecke module of divisors on the supersingular points on $X_0(N)$ in characteristic p is included for the first time.

Incidence Structures

- *Partitions and Tableaux:* An extensive range of facilities have been installed for calculating with Young tableaux. This work underpins a module for symmetric functions due for release shortly. The tableau machinery is based in part on the *Symmetrica* package.

- *Graphs:* Magma now supports two equivalent graph representations: A graph as an adjacency matrix (the dense representation) or as an adjacency list (the sparse representation). Some existing Magma functions –which up to now only dealt with the dense representation– have been rewritten for the sparse representation. Any conversion from one representation to the other that may be required by internal specifications will be automatic, that is, without user intervention.

- *Graphs:* A major new facility for graphs is the implementation of a linear-time algorithm due to Boyer and Myrvold for determining whether or not a graph is planar. If the graph is planar, a planar embedding may be obtained. This machinery utilizes the sparse representation.

Incidence Geometries

- *Incidence Geometries:* Magma now provides facilities for computing shadows and shadowspaces starting from an incidence geometry. Moreover, if an incidence geometry is a graph, it can be converted into an object of type Graph.

- *Coset Geometries:* The diagram of a coset geometry may now be computed (which is much faster than computing the diagram of an incidence geometry). Kernels, quotients, minimal parabolic subgroups are also available. Moreover, if a coset geometry is a graph, it can be converted into an object of type Graph.

3 Documentation

New chapters in the Handbook for V2.9 are:

- Braid Groups
- Lazy Power Series
- Class Field Theory
- Sparse Matrices
- The Module of Supersingular Points
- Partitions and Tableaux

4 Groups

4.1 General Groups

New Features:

- The facilities for random element generation now include the functions `RandomProcessWithValues`, `RandomProcessWithWords` and `RandomProcessWithWordsAndValues`. The `Values` option allows parallel computation of homomorphic images, while the `Words` option gives the output as a straight line program in the original generators.

4.2 Permutation Groups

New Features:

- The Schreier-Todd-Coxeter-Sims algorithm for computing the order of a permutation group has been revised, yielding an improved performance of various algorithms for permutation groups.
- The choice of algorithm used when verifying the correctness of a BSGS is changed, resulting in improved performance in degrees 100 to 4000 particularly.
- The function `ExtraSpecialGroup(p, n)` now accepts a parameter `Type`, facilitating the creation of both isomorphism types of extra special groups.
- A fast `IntersectionWithNormalSubgroup` function, based on an algorithm of Cooperman, Finkelstein & Luks, has been implemented and is used in socle computations.
- The `SocleQuotient` algorithm has been improved by reducing the size of the coset enumerations performed.
- New algorithms for `ChiefSeries` and `ChiefFactors` have been implemented.
- Heavier use is made of straight line programs in defining strong generators and computing homomorphisms.
- Computation of `MinimalPartitions` (including primitivity testing) now uses the nearly-linear time algorithm of Schönert and Seress, which is considerably faster than the previous method, particularly at high degree.

– Computations using the `BlocksAction` homomorphisms have been revised and greatly improved.

Bug fixes:

– A collection of bugs in the filtering of the groups in the various `Subgroups` functions have been fixed.

– Bugs in the construction of socles of primitive permutation groups have been fixed. A bug in computing composition factors and testing simplicity, related to the socle problem, has also been fixed. These bugs, as far as we know, were restricted to the primitive subgroups of the degree 21^5 representation of $L_2(7).2 \wr S_5$.

– Excessive recursion depth in computing the socle of an intransitive group has been addressed and reduced.

– Unnecessary calls to BSGS verification routines in several constructions have been addressed and reduced.

4.3 General Matrix Groups

Changes:

– The Schreier-Todd-Coxeter-Sims algorithm for computing the order of a matrix group has been revised, yielding an improved performance of various algorithms for matrix groups.

– Heavier use is made of straight line programs in defining strong generators and computing homomorphisms.

– `DerivedSeries` and `PCGroup` algorithms are improved.

– The `LineOrbits` function now returns orbits which are made up of 1-dimensional spaces, rather than vectors.

New features:

– Algorithms for computing the maximal p-quotient and the maximal nilpotent quotient of a matrix group have been provided.

– New features: `ChiefSeries`, `ChiefFactors`, `ElementaryAbelianSeries`, `RadicalQuotient` and `Radical` have been included for matrix groups with BSGS, provided the base ring is a field or integral domain.

Bug fixes:

– A bug allowing the user to attempt iteration over an infinite group has been fixed.

– Unnecessary calls to BSGS verification routines in several constructions have been addressed and reduced. This work extends to unnecessary checks of finiteness in several situations.

4.4 Finitely Presented Groups [HB 20, 21]

Changes:

- The kernels of maps returned by the various quotient algorithms are now computed only when these kernels are accessed. This significantly speeds up the construction of these quotient maps.
- The handling of subgroup relationships has been revised completely. A group H can now be recognised as subgroup of groups other than a group G as subgroup of which H was defined or of supergroups of G. This greatly enhances the possibilities of further computations with such subgroups and remedies a variety of problems.
- The function `ExtraSpecialGroup(GrpFP, p, n)` now accepts a parameter `Type`, facilitating the creation of both isomorphism types of extra special groups.
- Like for other group categories, the function `CosetAction(G, H)` now returns the kernel of the coset action as third return value, provided the index in G in small enough to represent this group.

New features:

- A new function `DerivedGroup(G)` computing the first derived subgroup of a finitely presented group G has been provided.
- A new function `Pullback(f, N)` has been added. Given a map f from an fp-group G to an $\mathbf{F}_p[G]$-module M and a submodule N of M, this function uses a fast pullback technique to compute the preimage of N under f. This function is helpful for constructing normal subgroups of G. A similar approach is now used in suitable situations for computing preimages using the preimage constructor.
- New functions `HasComputableAbelianQuotient(G)` and `HasInfiniteComputableAbelianQuotient(G)` have been provided.

Bug fixes:

- A problem with the function `Darstellungsgruppe(G)` has been solved. This function now returns the correct answers also for groups G with an infinite abelian quotient.
- Problems with the functions `AbelianQuotientInvariants(G)` and `ReduceGenerators(G)` have been fixed. Moreover, several other bugs and memory leaks have been removed.

4.5 Polycyclic Groups

Changes:

- The collection algorithm for computing in polycyclic groups has been improved. Magma now uses a hybrid algorithm written by Volker Gebhardt. This algorithm combines good asymptotic complexity in the exponents of group elements with efficient handling of small exponents. The gain in speed compared to the algorithm used in the previous release of Magmais about a factor of 5. Most operations on polycyclic groups profit from this speed up.
- The function `ExtraSpecialGroup(GrpGPC, p, n)` now accepts a parameter `Type`, facilitating the creation of both isomorphism types of extra special groups.

New features:

- Three new functions `WreathProduct` have been added, which compute the wreath product of two polycyclic groups with a specified action.

4.6 Finite Soluble Groups

Changes:

- The function `ExtraSpecialGroup(GrpPC, p, n)` now accepts a parameter `Type`, facilitating the creation of both isomorphism types of extra special groups.

4.7 Groups Defined by Rewrite Systems

New features:

- A new function `ElementToSequence(e)` for converting an element of a group G in the category `GrpRWS` to an integer sequence has been added. Conversely, an integer sequence can now be coerced to a word in the generators of G.
- Coercion of elements of a group G in the category `GrpRWS` to a member of one of the the other `KBMAG` categories `GrpAtc` and `MonRWS` is now possible.
- Homomorphisms with domain or codomain in the category `GrpRWS` are now supported.

4.8 Automatic Groups

New features:

- A new function `ElementToSequence(e)` for converting an element of a group G in the category `GrpAtc` to an integer sequence has been added. Conversely, an integer sequence can now be coerced to a word in the generators of G.
- Coercion of elements of a group G in the category `GrpAtc` to a member of one of the the other `KBMAG` categories `GrpRWS` and `MonRWS` is now possible.
- Homomorphisms with domain or codomain in the category `GrpAtc` are now supported.

4.9 Braid Groups (New) [HB31]

This new category comprises the family of all braid groups. Note that this special class of finitely presented groups has a solvable word problem. Recently, braid groups have received some interest as possible sources of cryptosystems.

Algorithms for fast element arithmetic and normal form computations for elements have been implemented. Homomorphisms whose domain or codomain is a braid group are supported.

4.9.1 Construction and Arithmetic

- Definition of a braid group on n strings.
- Product, inverse, conjugate for elements.
- Element normal form and equality testing.
- Generation of pseudo-random elements.

4.9.2 Homomorphisms

– Construction and evaluation of a homomorphism whose domain or codomain is a braid group.

4.10 Subgroups of $PSL(2, R)$

New features:

– For a point z in the upper half complex plane union cusps, the function `EquivalentPoint(z)` returns a point x and a matrix g in PSL(2,Z) such that x is in the fundamental domain for PSL(2,Z) given by -1/2 ¡ z ¡= 1/2, —z— ¿= 1, and g*z = x.

– For a point z in the upper half plane union cusps, and for G a congruence subgroup, the function `Stabilizer(z,G)` returns a generator of the stablizer subgroup of z in G.

– The function `IsEquivalent(G,a,b)` for G a congruence subgroup has been extented to test the equivalence of a and b under the action of G, for a and b pairs of cusps.

– The function `UpperHalfPlaneWithCusps` can now also be invoked with `UpperHalfPlane` and `UpperHalfPlaneUnionCusps`.

4.11 Databases of Groups [HB33]

Changes:

– A new version of the Small Groups Library written by Besche, Eick and O'Brien has been installed. This database now contains the following groups.

  – All groups of order up to 2 000, excluding the groups of order 1 024.

  – The groups of orders 5^5 and 7^4.

  – The groups whose order is a product of at most 3 distinct primes.

  – The groups of order $q^n p$, where q^n is a prime-power dividing 2^8, 3^6, 5^5 or 7^4 and p is a prime different to q.

– The transitive groups database now extends to degree 30, incorporating Alexander Hulpke's latest results.

Bug Fixes:

– The outer automorphism of J_2 has been included in the almost simple groups database, fixing the the computation of subgroups and automorphisms for groups where J_2 appears as a composition factor.

5 Semigroups and Monoids

5.1 Monoids Defined by Rewrite Systems

New features:

– A new function `ElementToSequence(e)` for converting an element of a monoid M in the category `MonRWS` to an integer sequence has been added. Conversely, an integer sequence can now be coerced to a word in the generators of M.

– Coercion of elements of a monoid M in the category `MonRWS` to a member of one of the the other `KBMAG` categories `GrpAtc` and `GrpRWS` is now possible.

– Homomorphisms with domain or codomain in the category `MonRWS` are now supported.

6 Basic Rings

6.1 Integer Ring

New features:

– New functions `EulerPhiInverse` and `FactoredEulerPhiInverse` to compute all integers whose Euler-ϕ value is a given integer (including factored versions).

6.2 Finite Fields

New features:

– When determining the default primitive element of a finite field, if the default generator is not primitive, then small simple elements are now first tried (instead of random elements).

– Discrete logarithms in characteristic 2 are now performed by Emmanuel Thomé's implementation of Coppersmith's index-calculus algorithm.

7 Extensions of Rings

7.1 Algebraic Number Fields

General New Features:

– Number fields can be created as a splitting field of a univariate integer polynomial.

– Places and divisors for number fields

– A large number of functions dealing with abelian extensions of number fields have been added to the Class Field Theory package.

7.1.1 Elements, Ideals and Quotients

New features:

– A new parameter has been added to the class group computation to allow for a different set of random relations for large fields.

– Decopomposition type for prime ideal decomposition has been added.

7.2 Cyclotomic Fields

New Features:

- With this release it is possible to create sparse cyclotomic fields that allow arithmetic in much larger fields than in the dense model.

7.3 Abelian Extensions

New Features:

- Several new functions that exploit the Galois-module structure of the ray-class-groups.
- Functions to convert number fields into abelian extensions.
- Automorphism groups of an extension. A "guess" can be computed without the knowledge of defining equations.

7.4 Algebraic Function Fields

Extensions of Algebraic Function Fields can be made and a range of functions can be applied to fields created as such.

Removals and Changes:

- Representation matrices of function field elements and elements of their orders have been transposed to be consistent with other Magma matrices.
- It has been ensured that the first argument to the `Decomposition` functions is always a ring.

New Features:

- The `TateLichtenbaumPairing` of two (non relative) divisors can be formed.
- A `Parameterization` of a function field, given a divisor to determine the pole, can be gained.
- Magma level printing has been implemented for fields, orders, elements, ideals, places and divisors.
- Extensions of algebraic function fields and their orders can be formed. Extensions of global algebraic function fields can only be formed by setting the optional parameter `Check` to `false` due to a missing test for polynomial irreducibility over global function fields.
- Maximal Orders, both finite and infinite, of extensions of algebraic function fields can be found.
- Most related structures of extensions can be found. The structures which cannot are `ExactConstantField` of a field and `Reduce` of an order.
- Invariants such as `Characteristic`, `Degree`, `DefiningPolynomial`, `Basis` and `Discriminant` can be determined for relative extensions and their orders.
- `UnitRank` of a finite maximal relative order of a global function field can be computed.
- Homomorphisms of all function fields can now be formed using the `hom<|>` constructor.
- Elements of relative extensions and their orders can be created, converted to a sequence and can have the usual arithmetic operators applied to them.
- Elements of extensions can be tested for equality and belonging to a certain ring or field. The predicates `IsDivisibleBy`, `IsZero`, `IsOne`, `IsMinusOne`, `IsNilpotent`, `IsIdempotent`, `IsUnit`, `IsZeroDivisor`, `IsRegular`, `IsIrreducible` and `IsPrime` can be tested on them.
- The `Norm`, `Trace`, `Minimal` and `CharacteristicPolynomials` and `RepresentationMatrix` of an element of an extension can be calculated. `Numerator` and `Denominator` with respect to a given order can also be determined.
- Elements of extensions can be made from and converted to a `ProductRepresentation` and expressed as a `RationalFunction`.
- Ideals of extensions can be created from a list of generators or as the product of an element and an order. Arithmetic is possible for these new ideals as well as tests for equality and `IsOne`, `IsZero`, `IsIntegral` and `IsPrime`.
- The intersection, `GCD` and `LCM` of two ideals of extensions can be found. The order the ideal is of can be reported and the `Denominator` calculated.
- The `MultiplicatorRing`, `Minimum`, `Norm` and `Generators` of an ideal of an extension can be found. A `TwoElement` presentation can also be retrieved. Relative ideals can be split into an integral ideal and a denominator using `IntegralSplit`.

– The `Basis`, `BasisMatrix` and `TransformationMatrix` of a relative ideal can be gained.

– Ideals of extensions can be factorized. `Decomposition` of prime ideals can be performed. The valuation of an ideal at a prime ideal and valuation of elements at a prime ideal can be calculated. The Ramification and Inertia Degrees of such ideals can be calculated. The `ResidueClassField` of an ideal can be returned.

– The `Module` and `Relations` functions can be applied to elements of extensions of algebraic function fields though there are restrictions as to the ring the resulting module is over.

– Factorization of polynomials over extensions of algebraic function fields can be accomplished.

Bug Fixes:

– Printing of elements has been tidied.

– A bug in `Expand` has been fixed.

– `GCD` and `LCM` of divisors was sometimes missing some places in the support of the result – this has been fixed.

7.5 Newton Polygons

Bug Fixes:

– Roots which were not found of a polynomial over a series ring are now found. This only affected some of the polynomials whose newton polygons had only one face.

7.6 Lazy Series Rings (New) [HB62]

Power series rings containing only elements of infinite precision have been developed. The series in these rings know how to compute their coefficients but their coefficients are unknown till asked for.

New Features:

– Lazy power series rings of any rank over any ring can be created. Once created another lazy series ring can be created from it having the same rank but a different coefficient ring.

– The dot notation can be used to retrieve the variables of a series ring and names can be assigned to these variables.

– The coefficient ring and rank are accessible from the lazy series ring.

– Lazy series can be created from constants in the coefficient ring, from series in other lazy series rings with the same rank and from polynomials and rational functions. Sequences of coefficient ring elements can also be coerced into a lazy series ring to create a series. Series can be created using the `elt` constructor where the right hand side is a map taking exponents to coefficients.

– Further series can be created by addition, subtraction, multiplication and taking powers (including inverses) of existing series.

– The sum and product of a series and a ring element can be formed. It is also possible to multiply a series by a monomial by providing the exponents of the variables in the monomial.

– Single coefficients of a series can be calculated as well as coefficients of monomials up to some total exponent degree. These coefficients are ordered with respect to the ordering of their monomials which is the same as the default ordering of monomials in multivariate polynomial rings. Valuation with respect to this ordering can also be obtained and leading coefficients and terms can be obtained. `PrintToPrecision` will print the series as a linear combination of coefficients and monomials in this order. `PrintTermsOfDegree` will print the terms of a given degree in the same way as `PrintToPrecision`.

– Coefficients can also be returned using the "odometer" ordering. An upper limit for the exponents of each variable must be provided. Starting with the constant term, the exponent of the last variable will be incremented to its limit, then the exponent of the second last incremented and the same done with the last exponent and so on. The index of a coefficient in the returned sequence given the exponent of the monomial can be found.

– Lazy series can be tested for being equal, zero, one, minus one or a unit. This will look at how the series was created. For two series to be equal they must have been created the same or be known to have finitely many non–zero coefficients which are the same. `IsWeaklyZero` and `IsWeaklyEqual` can be used giving a degree for which all terms of lesser degree must be zero or equal.

– Lazy series can be created as the (nth) derivative or integral of another series, as the polynomial coefficient of a series over a polynomial ring, as the evaluation of a series at (an)other series and as the square root of a univariate series.

8 Linear Algebra and Module Theory

8.1 Sparse Matrices (New)

A special type for sparse matrices is now provided so that the user can build up such matrices and then apply some non-trivial algorithms to them. An extended example in the Handbook implements the basic linear sieve for discrete logarithms in the Magma language, thus demonstrating how one can use the sparse matrix facilities when implementing index-calculus methods.

Features:

– Creation of sparse matrices in compact form.

– Creation of trivial sparse matrices followed by dynamic expansion.

– Basic properties (density, etc.).

– Conversion between sparse and normal (dense-representation) matrices.

– Multiplication of dense vectors by sparse matrices.

– Non-trivial invariants of sparse matrices: nullspace, rank and elementary divisors (equivalent to Smith form).

– Computation of non-zero solution vector for sparse systems arising in index-calculus algorithms (Structured Gaussian elimination and Lanczos algorithms).

8.2 Modules over Dedekind domains

Modules over Dedekind domains used to be only Modules over maximal orders of Algebraic Number Fields. It is now possible to create such modules over maximal orders of Algebraic Function Fields as well.

New Features:

- Modules over maximal orders of Algebraic Function Fields can be created. The whole functionality of modules over maximal orders of Algebraic Number Fields carries over to the new modules.

Bug Fixes:

- A bug in creating a sub module of a free module has been fixed.

9 Algebraic Geometry

9.1 Schemes

Removals and Changes:

- Compositions of maps may be stored as a composition rather than as an expanded map.
- Schemes with length zero can no longer be created. This also eliminates projective spaces with length one, that is, those spaces whose affine patches have length zero.

New Features:

- A number of alternative equations can be given when creating maps between schemes.

9.1.1 Function Fields and Divisors on Curves

Changes:

- The function `Place(C, m)` has been renamed `HasPlace`.

9.2 Elliptic Curves

9.2.1 General Elliptic Curves

Removals and Changes:

- Maps to or from Elliptic Curves are now the same as those between general schemes which have a special map type.

9.3 Hyperelliptic Curves

9.3.1 Hyperelliptic Curves

Removals and Changs:

- Maps to or from Hyperelliptic curves are now the same as maps between general schemes which have a special map type.
- Parents of maps to or from Hyperelliptic curves now inherit from `PowMap` and as such only allow coercion of maps into them. Coercions of `<t, e, u>` information are no longer available but similar functionality can be accessed using `Transformation`.

9.4 The Module of Supersingular Points (New)

A module for computing with the Hecke module of divisors on the supersingular points on $X_0(N)$ in characteristic p is now included. This is the free abelian group on the supersingular elliptic curves in characteristic p enhanced with level N structure. We compute this module using the Method of Graphs of Mestre and Oesterlé and the Brandt modules algorithm.

- Computation of Hecke operators and Atkin-Lehner involution.
- Decomposition into invariant subspaces.
- The monodromy pairing.

9.5 Modular Forms

New features:

- Computation of Atkin-Lehner operators.

10 Incidence Structures

10.1 Partitions and Tableaux

- Number of partitions of n
- Enumeration of restricted and unrestricted partitions
- Conjugate partitions
- Longest increasing sequences within words of integers
- Lexicographical ordering of words
- Creatation of skew or non-skew tableaux over the integers or arbitrary label sets
- Enumeration of tableaux

– Random tableaux

– Properties: shape, skew-shape, weight, hooklength's, content, row/column words

– Operations: diagonal sum, product, conjugate, jeu de taquin, row insertion, inverse jeu de taquin, inverse row insertion

– RSK correspondence, inverse RSK correspondence

– Enumeration of tableaux having specified size and alphabet

10.2 Graphs

Changes:

– The function `Bicomponents` now returns the bicomponents as a sequence of subsets of the graph's vertex set, instead of a sequence of subgraphs as previously.

– The functions `DFSTree(`v`)` and `BFSTree(`v`)` may now apply to disconnected graphs too: the tree returned is simply the search tree rooted at vertex v.

– The functions `IsDistanceTransitive`, `IsTransitive`, `IsPrimitive`, `IsSymmetric`, `IsEdgeTransitive`, `EdgeGroup`, `OrbitsPartition`, `IsIsomorphic`, and `CanonicalGraph` no longer accept user parameters. Whenever parameters are needed to drive the computation of the graph's automorphism group, users may set the appropriate parameters when calling `AutomorphismGroup`.

New Features:

– It is now possible to construct graphs with a sparse representation: That is, the graph is internally represented by means of its adjacency list.

  If no specification is given at the time of constructing the graph, or if the called Magma function does not allow for such a specification to be given, the graph is *always* represented by means of its adjacency matrix (the dense representation).

  Not all existing Magma functions have been –yet– rewritten with the new sparse representation in mind. Should internal specifications of a Magma function require a different graph representation than the current one, then the required conversion takes place automatically, *without* user intervention.

  The primary purpose of introducing the sparse representation is the implementation of a planarity tester.

– A linear planarity tester and a linear obstruction isolator due to Boyer and Myrvold has been implemented. One tests if a graph G is planar by calling `IsPlanar(`G`)`. This functions also returns the obstruction as a subgraph of G should G be non-planar.

  If G is planar, additional functionalities allow the computation of G's faces (`Faces(`G`)`), and of G's embedding (`Embedding(`G`)`).

  The planarity tester may be applied to any undirected graph, connected or not.

– It is now possible to determine the strongly connected components of a digraph (`StronglyConnectedComponents`).

Bug Fixes:

– The Magma interface for the `nauty` program that computes the automorphism group of a graph `AutomorphismGroup` has been improved so that the efficiency of the Magma call to `nauty` is as close as possible to the efficiency of a call to `nauty` outside of Magma's interface.

– In this same interface, a memory allocation problem has been fixed. Because of this allocation problem, `AutomorphismGroup` could previously return wrong answers.

– The function `StronglyRegularGraphsDatabase` now reads the catalogue of strongly regular graphs correctly.

11 Incidence Geometry

11.1 Incidence Geometries

New Features:

– The function `IsGraph(D)` permits to check if the incidence geometry D is a graph.

– The function `Graph(D)` permits to convert the incidence geometry D into an object of type `Graph` provided D is a rank two geometry such that all elements of one type are incident with exactly two elements of the other.

– The function `Shadow(D, I, F)` computes the I-shadow of the flag F in the incidence geometry D.

– The `ShadowSpace(D, I)` function computes the shadow space of type I of the incidence geometry D.

11.2 Coset Geometries

New Features:

– Some functions have been added to access the parabolic subgroups of a coset geometry : `Group(D)` returns the group from which D is constructed, `MinimalParabolics(D)` or `MinParabolics(D)` returns an indexed set containing all the minimal parabolic subgroups of D.

– The function `Diagram(C)` computes the diagram of the coset geometry C. This algorithm is much faster than the one for incidence geometries since it uses lots of group theory machinery in the computation.

– The function `IsGraph(C)` permits to check if the coset geometry C is a graph.

– The function `Graph(C)` permits to convert the coset geometry C into an object of type `Graph` provided C is a rank two geometry such that all elements of one type are incident with exactly two elements of the other.

– The full kernel of the coset geometry D can be computed using the `Kernel(D)` function and the i-kernels of D using the `Kernels(D)` function.

– The function `Quotient(D,K)` returns the quotient of the coset geometry D by a subgroup K provided that K is a subgroup of the kernel of D.