Sage Reference Manual

Release 6.3

The Sage Development Team

August 11, 2014

CONTENTS

This is a thematic index of all of Sage's features. It also contains many examples that illustrate their use, all of them systematically tested with each release.

Enjoy Sage!

CHAPTER

ONE

TABLE OF CONTENTS

1.4 Rings, Fields, Algebras

1.5 Groups, Monoids, Matrices, Modules

1.6 Geometry and Topology

1.7 Number Theory, Algebraic Geometry

- Modular Symbols
- Modular Forms
- Modular Abelian Varieties
- Miscellaneous Modular-Form-Related Modules

1.8 Miscellaneous Mathematics

- Games
- Symbolic Logic
- SAT solvers
- Cryptography
- Numerical Optimization
- Probability
- Statistics
- Quantitative Finance
- Coding Theory
- Game Theory

1.9 Doctesting, Interfaces, Databases, Miscellany

- Doctesting
- Development Scripts
- Interpreter Interfaces
- C/C++ Library Interfaces
- Databases
- Parallel Computing
- Miscellaneous

1.10 Other

1.10.1 SAGE's To Do list

There is still some work to do :-) :

Warning: This list is currently very incomplete as most doctests do not use the `.. todo::` markup.

Todo
Rewrite the hand-written TODOs by using the correct `.. todo::` markup.

The combined to do list is only available in the html version of the reference manual.

Todo

Rewrite the hand-written TODOs by using the correct `.. todo::` markup.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/src/doc/en/reference/todolist.rst, line 13.)

Todo

An example illustrating unitary flag.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/algebras/finite_dimensional_algebras/finite_dimensional_algebra_morphism.py:docstring of sage.algebras.finite_dimensional_algebras.finite_dimensional_algebra_morphism.FiniteDimensionalAlgebraMorphism, line 35.)

Todo

- Coercion doesn't work yet, there is some cheating about assumptions
- The optional argument `check` controls checking the degeneracy conditions. Furthermore, the default values interfere with non-degeneracy conditions.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/algebras/free_algebra.py:docstring of sage.algebras.free_algebra.FreeAlgebra_generic.g_algebra, line 4.)

Todo

Implement multi-parameter Iwahori-Hecke algebras together with their Kazhdan-Lusztig bases. That is, Iwahori-Hecke algebras with (possibly) different parameters for each conjugacy class of simple reflections in the underlying Coxeter group.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/algebras/iwahori_hecke_algebra.py:docstring of sage.algebras.iwahori_hecke_algebra.IwahoriHeckeAlgebra, line 305.)

Todo

When given "generic parameters" we should return the generic Iwahori-Hecke algebra with these parameters and allow the user to work inside this algebra rather than doing calculations behind the scenes in a copy of the generic Iwahori-Hecke algebra. The main problem is that it is not clear how to recognise when the parameters are "generic".

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/algebras/iwahori_hecke_algebra.py:docstring of sage.algebras.iwahori_hecke_algebra.IwahoriHeckeAlgebra, line 312.)

Todo

- "filtered" DFTs
- more idfts
- more examples for probability, stats, theory of FTs

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/gsl/dft.py:docstring of sage.gsl.dft, line 52.)

Todo

Read the parent of the elements of S; if **Q** or **C** leave as is; if AbelianGroup, use abelian_group_dual; if some other implemented Group (permutation, matrix), call .characters() and test if the index list is the set of conjugacy classes.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/gsl/dft.py:docstring of sage.gsl.dft.IndexedSequence.dft, line 41.)

Todo

Add an example.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/additive_magmas.py:docstring of sage.categories.additive_magmas.AdditiveMagmas.ParentMethods.summati line 34.)

Todo

Add an example.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/additive_magmas.py:docstring of sage.categories.additive_magmas.AdditiveMagmas.ParentMethods.summati line 34.)

Todo

add a description of this category

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/affine_weyl_groups.py:docstring of sage.categories.affine_weyl_groups.AffineWeylGroups, line 3.)

Todo

should return an enumerated set, with iterator, ...

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/affine_weyl_groups.py:docstring of sage.categories.affine_weyl_groups.AffineWeylGroups.ParentMethods.aff line 13.)

Todo

- Add support for non commutative rings (this is currently not supported by the subcategory `AlgebraModules`).
- Make `AlgebraIdeals(R)`, return `CommutativeAlgebraIdeals(R)` when `R` is commutative.
- If useful, implement `AlgebraLeftIdeals` and `AlgebraRightIdeals` of which `AlgebraIdeals` would be a subcategory.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/algebra_ideals.py:docstring of sage.categories.algebra_ideals.AlgebraIdeals, line 8.)

Todo

Should R be a commutative ring?

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/algebras.py:docstring of sage.categories.algebras.Algebras, line 14.)

Todo

Improve this explanation.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/category.py:docstring of sage.categories.category.Category._without_axioms, line 8.)

Todo

Add an optional argument to allow for:

```
sage: Realizations(A, category = Blahs()) # todo: not implemented
```

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/category.py:docstring of sage.categories.category.Category.Realizations, line 39.)

Todo

Get a consistent hierarchy of homset categories. Currently, it is built in parallel to that of their base categories (which is plain wrong!!!)

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/category.py:docstring of sage.categories.category.HomCategory, line 3.)

Todo

Further remove the base ring (see also trac ticket #15801).

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/category.py:docstring of sage.categories.category.category_graph, line 15.)

Todo

Specify whether or not one should systematically use @cached_method in the definition of the axiom. And make sure all the definition of axioms in Sage are consistent in this respect!

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/category_with_axiom.py:docstring of sage.categories.category_with_axiom, line 376.)

Todo

We could possibly define an @axiom decorator? This could hide two little implementation details: whether or not to make the method a cached method, and the call to _with_axiom(...) under the hood. It could do possibly do some more magic. The gain is not obvious though.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/category_with_axiom.py:docstring of sage.categories.category_with_axiom, line 382.)

Todo

Explore ways to get rid of this global `all_axioms` tuple, and/or have automatic registration there, and/or having a register_axiom(...) method.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/category_with_axiom.py:docstring of sage.categories.category_with_axiom, line 421.)

Todo

Other categories that would be better implemented via an axiom depending on a join category include:

- `Algebras`: defining an associative unital algebra as a ring and a module satisfying the suitable compatibility axiom between inner multiplication and multiplication by scalars (bilinearity). Of course this should be implemented at the level of `MagmaticAlgebras`, if not higher.
- `Bialgebras`: defining an bialgebra as an algebra and coalgebra where the coproduct is a morphism for the product.
- `Bimodules`: defining a bimodule as a left and right module where the two actions commute.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/category_with_axiom.py:docstring of sage.categories.category_with_axiom, line 488.)

Todo

- Design and implement an idiom for the definition of an axiom by a join category.
- Or support more advanced joins, through some hook or registration process to specify that a given category *is* the intersection of two (or more) categories.
- Or at least improve the above workaround to avoid the last issue; this possibly could be achieved using a class `Magmas.Distributive` with a bit of `__classcall__` magic.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/category_with_axiom.py:docstring of sage.categories.category_with_axiom, line 505.)

Todo

The above example violates the specifications (a category should be modelled by at most one class), so it's appropriate that it fails. Yet, the error message could be usefully complemented by some hint at what the source of the problem is (a category implemented in two distinct classes). Leaving a large enough piece of the backtrace would be useful though, so that one can explore where the issue comes from (e.g. with post mortem debugging).

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/category_with_axiom.py:docstring of sage.categories.category_with_axiom, line 1086.)

Todo

The following specifications would be desirable but are not yet implemented:

- A functorial construction category (Graded, CartesianProducts, ...) having a `Category_singleton` as base category should be a `CategoryWithAxiom_singleton`.

  Nothing difficult to implement, but this will need to rework the current "no subclass of a concrete class" assertion test of `Category_singleton.__classcall__()`.

- Similarly, a covariant functorial construction category having a `Category_over_base_ring` as base category should be a `Category_over_base_ring`.

The following specification might be desirable, or not:

- A join category involving a `Category_over_base_ring` should be a `Category_over_base_ring`. In the mean time, a `base_ring` method is automatically provided for most of those by `Modules.SubcategoryMethods.base_ring()`.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/category_with_axiom.py:docstring of sage.categories.category_with_axiom, line 1248.)

Todo

Detail this a bit. What could typically go wrong is a situation where, for some category `C1`, `C1.A()` specifies a category `C2` as super category such that `C2.A()` specifies `C3` as super category such that ...; this would clearly cause an infinite execution. Note that this situation violates the specifications since `C1.A()` is supposed to be a subcategory of `C2.A()`, ... so we would have an infinite increasing chain of constructible categories.

It's reasonnable to assume that there is a finite number of axioms defined in the code. There remains to use this assumption to argue that any infinite execution of the algorithm would give rise to such an infinite sequence.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/category_with_axiom.py:docstring of sage.categories.category_with_axiom, line 1403.)

Todo

Decide whether we care about this feature. In such a situation, we are not really defining a new axiom, but just defining an axiom as an alias for a couple others, which might not be that useful.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/category_with_axiom.py:docstring of sage.categories.category_with_axiom.Blahs.Blue_extra_super_categorie
line 15.)

Todo

Improve the infrastructure to detect and report this violation of the specifications, if this is easy. Otherwise, it's not so bad: when defining an axiom A in a category `Cs` the first thing one is supposed to doctest is that `Cs().A()` works. So the problem should not go unnoticed.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/category_with_axiom.py:docstring of sage.categories.category_with_axiom.Blahs.Blue_extra_super_categorie
line 22.)

Todo

add a demo of usual computations on Coxeter groups.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/coxeter_groups.py:docstring of sage.categories.coxeter_groups.CoxeterGroups, line 40.)

Todo

- Use the symmetric group in the examples (for nicer output), and print the edges for a stronger test.
- The constructed poset should be lazy, in order to handle large / infinite Coxeter groups.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/finite_coxeter_groups.py:docstring of sage.categories.finite_coxeter_groups.FiniteCoxeterGroups.ParentMeth
line 44.)

Todo

- Use the symmetric group in the examples (for nicer output), and print the edges for a stronger test.
- The constructed poset should be lazy, in order to handle large / infinite Coxeter groups.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/finite_coxeter_groups.py:docstring of sage.categories.finite_coxeter_groups.FiniteCoxeterGroups.ParentMeth
line 70.)

Todo

- Use the symmetric group in the examples (for nicer output), and print the edges for a stronger test.
- The constructed poset should be lazy, in order to handle large / infinite Coxeter groups.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/finite_coxeter_groups.py:docstring of sage.categories.finite_coxeter_groups.FiniteCoxeterGroups.ParentMeth
line 70.)

Todo

`sage.combinat.debruijn_sequence.DeBruijnSequences` should not inherit from this class. If that is solved, then `FiniteEnumeratedSets` shall be turned into a subclass of `Category_singleton`.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/finite_enumerated_sets.py:docstring of sage.categories.finite_enumerated_sets.FiniteEnumeratedSets, line 25.)

Todo

make this optional

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/finite_semigroups.py:docstring of sage.categories.finite_semigroups.FiniteSemigroups, line 17.)

Todo

Get rid of this workaround once there is a more systematic approach for the alias `Modules(QQ)` -> `VectorSpaces(QQ)`. Probably the later should be a category with axiom, and covariant constructions should play well with axioms.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/graded_modules.py:docstring of sage.categories.graded_modules.GradedModules.extra_super_categories, line 18.)

Todo

- Check which methods would be better located in `Monoid.Algebras` or `Groups.Finite.Algebras`.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/groups.py:docstring of sage.categories.groups.Groups.Algebras, line 24.)

Todo

- Design decision: how much of the homset comes from the category of `X` and `Y`, and how much from the specific `X` and `Y`. In particular, do we need several parent classes depending on `X` and `Y`, or does the difference only lie in the elements (i.e. the morphism), and of course how the parent calls their constructors.
- Specify the protocol for the `_Hom_` hook in case of ambiguity (e.g. if both a parent and some category thereof provide one).

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/homset.py:docstring of sage.categories.homset.Hom, line 172.)

Todo

Refactor during the upcoming homset cleanup.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/homset.py:docstring of sage.categories.homset.Homset.element_class_set_morphism, line 8.)

Todo

- Clarify the distinction, if any, with `BiModules(R, R)`. In particular, if R is a commutative ring (e.g. a field), some pieces of the code possibly assume that M is a *symmetric 'R'-'R'-bimodule*:

$$r * x = x * r \qquad \forall r \in R \text{ and } x \in M$$

- Make sure that non symmetric modules are properly supported by all the code, and advertise it.
- Make sure that non commutative rings are properly supported by all the code, and advertise it.
- Add support for base semirings.
- Implement a `FreeModules(R)` category, when so prompted by a concrete use case: e.g. modeling a free module with several bases (using `Sets.SubcategoryMethods.Realizations()`) or with an atlas of local maps (see e.g. trac ticket #15916).

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/modules.py:docstring of sage.categories.modules.Modules, line 57.)

Todo

- Explain why this does not commute with `WithBasis()`
- Improve the support for covariant functorial constructions categories over a base ring so as to get rid of the `base_ring` argument.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/modules.py:docstring of sage.categories.modules.Modules.SubcategoryMethods.Graded, line 18.)

Todo

handle base being a category

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/modules.py:docstring of sage.categories.modules.Modules.SubcategoryMethods.base_ring, line 6.)

Todo

- Implement an optimized `_call_()` function.
- Generalize to a mapcoeffs.
- Generalize to a mapterms.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/modules_with_basis.py:docstring of sage.categories.modules_with_basis.DiagonalModuleMorphism, line 23.)

Todo

`End(X)` is an algebra.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/modules_with_basis.py:docstring of sage.categories.modules_with_basis.ModulesWithBasis, line 85.)

Todo

Should codomain be `self` by default in the diagonal and triangular cases?

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/modules_with_basis.py:docstring of sage.categories.modules_with_basis.ModulesWithBasis.ParentMethods.n line 189.)

Todo

Extract a method to linearize a multilinear morphism, and delegate the work there.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/modules_with_basis.py:docstring of sage.categories.modules_with_basis.ModulesWithBasis.TensorProducts.I line 99.)

Todo

This has nothing to do here!!! Should there be a library for pointwise operations on functions somewhere in Sage?

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/modules_with_basis.py:docstring of sage.categories.modules_with_basis.pointwise_inverse_function, line 21.)

Todo

shall we accept only permutations with finite support or not?

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/permutation_groups.py:docstring of sage.categories.permutation_groups.PermutationGroups, line 6.)

Todo

Give a concrete example, typically using `ElementWrapper`.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/primer.py:docstring of sage.categories.primer, line 276.)

Todo

Improve the printing of functorial constructions and joins to raise this potentially dangerous ambiguity.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/primer.py:docstring of sage.categories.primer, line 1451.)

Todo

Add an optional argument to allow for:

```
sage: Realizations(A, category = Blahs()) # todo: not implemented
```

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/realizations.py:docstring of sage.categories.realizations.Realizations, line 39.)

Todo

(see: http://trac.sagemath.org/sage_trac/wiki/CategoriesRoadMap)

- Make Rings() into a subcategory or alias of Algebras(ZZ);
- A parent P in the category `Rings()` should automatically be in the category `Algebras(P)`.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/rings.py:docstring of sage.categories.rings.Rings, line 27.)

Todo

Make `Schemes()` a singleton category (and remove `Schemes` from the workaround in `category_types.Category_over_base._test_category_over_bases()`).

This is currently incompatible with the dispatching below.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/schemes.py:docstring of sage.categories.schemes.Schemes, line 17.)

Todo

- Add more options for constructing subgraphs of the Cayley graph, handling the standard use cases when exploring large/infinite semigroups (a predicate, generators of an ideal, a maximal length in term of the generators)
- Specify good default layout/plot/latex options in the graph
- Generalize to combinatorial modules with module generators / operators

---

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/semigroups.py:docstring of sage.categories.semigroups.Semigroups.ParentMethods.cayley_graph, line 110.)

Todo

Draw the typical commutative diagram.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/sets_cat.py:docstring of sage.categories.sets_cat.Sets.SubcategoryMethods.Subquotients, line 27.)

Todo

use a more interesting example, like $\mathbf{Z}/n\mathbf{Z}$.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/sets_cat.py:docstring of sage.categories.sets_cat.Sets.SubcategoryMethods.Subquotients, line 121.)

Todo

Fix the failing test by making `C` a singleton category. This will require some fiddling with the assertion in `Category_singleton.__classcall__()`

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/sets_cat.py:docstring of sage.categories.sets_cat.Sets.WithRealizations.ParentMethods.Realizations, line 14.)

Todo

- This should be moved to `Sets().WithGrading()`.
- Should the grading set be a parameter for this category?
- Does the enumeration need to be compatible with the grading? Be careful that the fact that graded components are allowed to be finite or infinite make the answer complicated.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/categories/sets_with_grading.py:docstring of sage.categories.sets_with_grading.SetsWithGrading, line 72.)

Todo

Implement multi-parameter Iwahori-Hecke algebras together with their Kazhdan-Lusztig bases. That is, Iwahori-Hecke algebras with (possibly) different parameters for each conjugacy class of simple reflections in the underlying Coxeter group.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/algebras/iwahori_hecke_algebra.py:docstring of sage.algebras.iwahori_hecke_algebra.IwahoriHeckeAlgebra, line 305.)

Todo

When given "generic parameters" we should return the generic Iwahori-Hecke algebra with these parameters and allow the user to work inside this algebra rather than doing calculations behind the scenes in a copy of the generic Iwahori-Hecke algebra. The main problem is that it is not clear how to recognise when the parameters are "generic".

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/algebras/iwahori_hecke_algebra.py:docstring of sage.algebras.iwahori_hecke_algebra.IwahoriHeckeAlgebra, line 312.)

Todo

Do we want to implement the following syntactic sugar:

```
with t.clone() as tt:
    tt.labels[1,2] = 3 ?
```

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/abstract_tree.py:docstring of sage.combinat.abstract_tree.AbstractLabelledClonableTree.set_label, line 34.)

Todo

It is currently not possible to use `LabelledBinaryTree()` as a shorthand for `LabelledBinaryTree(None)` (in analogy to similar syntax in the `BinaryTree` class).

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/binary_tree.py:docstring of sage.combinat.binary_tree.LabelledBinaryTree, line 50.)

Todo

GUAVA commands:

- VandermondeMat
- GrayMat returns a list of all different vectors of length n over the field F, using Gray ordering.

Not in GAP:

- Rencontres numbers http://en.wikipedia.org/wiki/Rencontres_number

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/combinat.py:docstring of sage.combinat.combinat, line 93.)

Todo

Incorporate this method into the `_repr_` for finite Cartan type.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/crystals/alcove_path.py:docstring of sage.combinat.crystals.alcove_path.CrystalOfAlcovePathsElement.integer line 4.)

Todo

Better doctest

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/crystals/alcove_path.py:docstring of sage.combinat.crystals.alcove_path.CrystalOfAlcovePathsElement.is_adm line 47.)

Todo

- Vocabulary and conventions:
  - For a classical crystal: connected / highest weight / irreducible
  - ...
- Layout instructions for plot() for rank 2 types
- RestrictionOfCrystal

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/crystals/crystals.py:docstring of sage.combinat.crystals.crystals, line 96.)

Todo

FIXME:

- Do we want to specify the columns increasingly or decreasingly? That is, should this be `Tab(columns = [[1,3],[2,4]])`?
- Make this fully consistent with `Tableau()`!

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/crystals/tensor_product.py:docstring of sage.combinat.crystals.tensor_product.CrystalOfTableaux, line 72.)

Todo

Implement finite non-Desarguesian plane as in [We07] and Wikipedia article Non-Desarguesian_plane.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/designs/block_design.py:docstring of sage.combinat.designs.block_design, line 28.)

Todo

Implement DerivedDesign and ComplementaryDesign.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/designs/design_catalog.py:docstring of sage.combinat.designs.design_catalog, line 56.)

Todo

There is a slightly more general version of difference families where the stabilizers of the blocks are taken into account. A block is *short* if the stabilizer is not trivial. The more general version is called a *partial difference family*. It is still possible to construct BIBD from this more general version (see the chapter 16 in the Handbook [DesignHandbook]).

Implement recursive constructions from Buratti "Recursive for difference matrices and relative difference families" (1998) and Jungnickel "Composition theorems for difference families and regular planes" (1978)

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/designs/difference_family.py:docstring of sage.combinat.designs.difference_family.difference_family, line 136.)

Todo

The XML data from the designtheory.org database contains a wealth of information about things like automorphism groups, transitivity, cycle type representatives, etc, but none of this data is made available through the current implementation.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/designs/ext_rep.py:docstring of sage.combinat.designs.ext_rep, line 14.)

Todo

- A resolvable $OA(k,n)$ is equivalent to a $OA(k+1,n)$. Sage should be able to return resolvable OA, with sorted rows (so that building the decomposition is easy.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/designs/orthogonal_arrays.py:docstring of sage.combinat.designs.orthogonal_arrays, line 35.)

Todo

As soon as wilson construction accepts an empty master design we should remove this intermediate functions.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/designs/orthogonal_arrays_recursive.py:docstring of sage.combinat.designs.orthogonal_arrays_recursive.simpl line 9.)

Todo

extend this to m-Dyck words

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/dyck_word.py:docstring of sage.combinat.dyck_word.CompleteDyckWords_size.random_element, line 11.)

Todo

At the moment, the letters of the alphabets need to be hashable.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/finite_state_machine.py:docstring of sage.combinat.finite_state_machine.FiniteStateMachine.determine_alphab line 17.)

Todo

Do the iteration in place to save on copying time

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/integer_list.py:docstring of sage.combinat.integer_list.IntegerListsLex.count, line 7.)

Todo

Placeholder. Implement a proper check.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/integer_list.py:docstring of sage.combinat.integer_list.IntegerListsLexElement.check, line 4.)

Todo

Move this into Cython.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/integer_list.py:docstring of sage.combinat.integer_list.first, line 8.)

Todo

should the order of the arguments `n` and `weight` be exchanged to simplify the logic ?

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/integer_vector_weighted.py:docstring of sage.combinat.integer_vector_weighted.WeightedIntegerVectors, line 39.)

Todo

Integer vectors should accept `max_part` as a single argument, and the following should change:

```
sage: S = IntegerVectorsModPermutationGroup(PermutationGroup([[(1,2,3,4)]]), max_part=12); S.ambient
Integer vectors
```

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/integer_vectors_mod_permgroup.py:docstring of sage.combinat.integer_vectors_mod_permgroup.IntegerVecto line 10.)

Todo

To study this, it would be more natural to define interval-posets on arbitrary ordered sets rather than just on $\{1, 2, \ldots, n\}$.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/interval_posets.py:docstring of sage.combinat.interval_posets.TamariIntervalPoset.insertion, line 25.)

Todo

Functionality to add:

- plotter will not plot edge labels higher than 2; e.g. in BK puzzles, the labels are 1,..., n and so in 3-step examples, none of the edge labels with 3 appear
- we should also have a 3-step puzzle pieces constructor, taken from p22 of Arxiv math/0610538
- implement the bijection from puzzles to tableaux; see for example R. Vakil, A geometric Littlewood-Richardson rule, Arxiv math/0302294 or K. Purbhoo, Puzzles, Tableaux and Mosaics, Arxiv 0705.1184.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/knutson_tao_puzzles.py:docstring of sage.combinat.knutson_tao_puzzles, line 12.)

Todo

This construction holds more generally for prime powers q congruent to $3 \mod 4$. We should implement these but we first need to implement Quadratic character for $GF(q)$.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/matrices/hadamard_matrix.py:docstring of sage.combinat.matrices.hadamard_matrix.H1, line 5.)

Todo

This construction holds more generally for prime powers q congruent to $1 \mod 4$. We should implement these but we first need to implement Quadratic character for $GF(q)$.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/matrices/hadamard_matrix.py:docstring of sage.combinat.matrices.hadamard_matrix.H2, line 6.)

Todo

Fix the failing test by making `C` a singleton category. This will require some fiddling with the assertion in `Category_singleton.__classcall__()`

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/ncsf_qsym/generic_basis_code.py:docstring of sage.combinat.ncsf_qsym.generic_basis_code.BasesOfQSymO line 14.)

Todo

Generalize this to all graded vector spaces?

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/ncsf_qsym/generic_basis_code.py:docstring of sage.combinat.ncsf_qsym.generic_basis_code.BasesOfQSymO line 33.)

Todo

Generalize this to all graded vector spaces?

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/ncsf_qsym/generic_basis_code.py:docstring of sage.combinat.ncsf_qsym.generic_basis_code.BasesOfQSymO line 47.)

Todo

Despite the `__repr__`, this is NOT an endomorphism!

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/ncsf_qsym/generic_basis_code.py:docstring of sage.combinat.ncsf_qsym.generic_basis_code.GradedModules line 34.)

Todo

Despite the `__repr__`, this is NOT an endomorphism!

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/ncsf_qsym/generic_basis_code.py:docstring of sage.combinat.ncsf_qsym.generic_basis_code.GradedModules line 34.)

Todo

Despite the `__repr__`, this is NOT an endomorphism!

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/ncsf_qsym/generic_basis_code.py:docstring of sage.combinat.ncsf_qsym.generic_basis_code.GradedModules line 34.)

Todo

demonstrate how to customize the basis names

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/ncsf_qsym/ncsf.py:docstring of sage.combinat.ncsf_qsym.ncsf.NonCommutativeSymmetricFunctions, line 162.)

Todo

explain the other changes of bases!

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/ncsf_qsym/ncsf.py:docstring of sage.combinat.ncsf_qsym.ncsf.NonCommutativeSymmetricFunctions, line 213.)

Todo

- Bases: monomial, fundamental, forgotten, quasi_schur_dual simple() ? (<=> simple modules of HS_n; to be discussed with Florent)
- Multiplication in:
  - fundamental and monomial (cf. Lenny's code)
  - ribbon (from Mike's code)

- Coproducts (most done by coercions)
- some_elements in all bases
- Systematic coercion checks (in AlgebrasWithBasis().Abstract())

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/ncsf_qsym/ncsf.py:docstring of sage.combinat.ncsf_qsym.ncsf.NonCommutativeSymmetricFunctions, line 259.)

Todo

this could be generalized to any free algebra.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/ncsf_qsym/ncsf.py:docstring of sage.combinat.ncsf_qsym.ncsf.NonCommutativeSymmetricFunctions.Multipli line 14.)

Todo

Implement this directly on the monomial basis maybe? The $(0, I)$-matrices are a pain to generate from their definition, but maybe there is a good algorithm. If so, the above "further examples" should be moved to the M-method.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/ncsf_qsym/qsym.py:docstring of sage.combinat.ncsf_qsym.qsym.QuasiSymmetricFunctions.Bases.ElementMe line 184.)

Todo

Implement this directly on the monomial basis maybe? The $(0, I)$-matrices are a pain to generate from their definition, but maybe there is a good algorithm. If so, the above "further examples" should be moved to the M-method.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/ncsf_qsym/qsym.py:docstring of sage.combinat.ncsf_qsym.qsym.QuasiSymmetricFunctions.Bases.ElementMe line 184.)

Todo

The conversion from the M basis to the HWL basis is currently implemented in the naive way (inverting the base-change matrix in the other direction). This matrix is not triangular (not even after any permutations of the bases), and there could very well be a faster method (the one given by Hazewinkel?).

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/ncsf_qsym/qsym.py:docstring of sage.combinat.ncsf_qsym.qsym.QuasiSymmetricFunctions.HazewinkelLamb line 39.)

Todo

accept an *alphabet* as input

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/ncsf_qsym/qsym.py:docstring of sage.combinat.ncsf_qsym.qsym.QuasiSymmetricFunctions.Monomial.Eleme line 14.)

Todo

Reimplement like `remove_horizontal_border_strip` using `IntegerListsLex`

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/partition.py:docstring of sage.combinat.partition.Partition.add_horizontal_border_strip, line 15.)

Todo

Check in Knuth AOCP4.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/partition.py:docstring of sage.combinat.partition.Partitions_n.random_element_uniform, line 21.)

Todo

This docstring needs to be fixed. First, the definition does not match the implementation (or the examples). Second, this doesn't seem to be defined in [GarStan1984] (the descent monomial in their (7.23) is different).

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/permutation.py:docstring of sage.combinat.permutation.Permutation.descent_polynomial, line 25.)

Todo

- generalize this feature by accepting a family of operators as input
- move up in some appropriate category

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/posets/linear_extensions.py:docstring of sage.combinat.posets.linear_extensions.LinearExtensionsOfPoset.mar line 8.)

Todo

Should the vertices of the diagram have the poset as parent?

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/posets/posets.py:docstring of sage.combinat.posets.posets.FinitePoset.hasse_diagram, line 5.)

Todo

The current algorithm could be improvable. See trac ticket #13223.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/posets/posets.py:docstring of sage.combinat.posets.posets.FinitePoset.is_graded, line 13.)

Todo

- Is it acceptable to have those two features for a single method?

- In particular, we miss a short idiom to get the default linear extension

---

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/posets/posets.py:docstring of sage.combinat.posets.posets.FinitePoset.linear_extension, line 44.)

Todo

add tests as in combinat::rankers

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/ranker.py:docstring of sage.combinat.ranker.on_fly, line 27.)

Todo

Implement a direct action of e_0 without moving to KR crystals.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/rigged_configurations/kr_tableaux.py:docstring of sage.combinat.rigged_configurations.kr_tableaux.KirillovRe
line 3.)

Todo

Implement a direct action of ϵ_0 without moving to KR crystals.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/rigged_configurations/kr_tableaux.py:docstring of sage.combinat.rigged_configurations.kr_tableaux.KirillovRe
line 3.)

Todo

Implement a direct action of f_0 without moving to KR crystals.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/rigged_configurations/kr_tableaux.py:docstring of sage.combinat.rigged_configurations.kr_tableaux.KirillovRe
line 3.)

Todo

Compute ϕ_0 without moving to KR crystals.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/rigged_configurations/kr_tableaux.py:docstring of sage.combinat.rigged_configurations.kr_tableaux.KirillovRe
line 3.)

Todo

Implement f_0 without appealing to tensor product of KR tableaux.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/rigged_configurations/rigged_configuration_element.py:docstring of sage.combinat.rigged_configurations.rigged_configuration_element.RiggedConfigurationElement.e, line 10.)

Todo

Implement f_0 without appealing to tensor product of KR tableaux.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/rigged_configurations/rigged_configuration_element.py:docstring of sage.combinat.rigged_configurations.rigged_configuration_element.RiggedConfigurationElement.f, line 12.)

Todo

Convert this to using multiplicities m_i (perhaps with a dictionary?)?

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/rigged_configurations/rigged_partition.py:docstring of sage.combinat.rigged_configurations.rigged_partition, line 18.)

Todo

add a method `set_mutable()` as, say, for matrices

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/cartan_type.py:docstring of sage.combinat.root_system.cartan_type, line 205.)

Todo

add a method `set_mutable()` as, say, for matrices

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/cartan_type.py:docstring of sage.combinat.root_system.cartan_type.CartanType, line 207.)

Todo

add some reducible Cartan types (suggestions?)

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/cartan_type.py:docstring of sage.combinat.root_system.cartan_type.CartanTypeFactory.samples, line 50.)

Todo

Add the picture here, once root system plots in the weight lattice will be implemented. In the mean time, the reader may look up the dual picture on Figure 2 of [HST09] which was produced by MuPAD-Combinat.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/cartan_type.py:docstring of sage.combinat.root_system.cartan_type.CartanType_affine.translation line 131.)

Todo

Add tests for the above assumptions, and also that the classical operators $T_1, \ldots, T_n$ from T and T_Y coincide.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/hecke_algebra_representation.py:docstring of sage.combinat.root_system.hecke_algebra_representation.CherednikOperatorsEigenvectors, line 39.)

Todo

Add an example where $T_i \neq T_i^{-1}$

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/hecke_algebra_representation.py:docstring of sage.combinat.root_system.hecke_algebra_representation.HeckeAlgebraRepresentation.Tw_inverse, line 5.)

Todo

Add more tests

Add tests in type BC affine where the null coroot $\delta^\vee$ can have non trivial coefficient in term of α_0

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/hecke_algebra_representation.py:docstring of sage.combinat.root_system.hecke_algebra_representation.HeckeAlgebraRepresentation.Y_lambdacheck, line 66.)

Todo

At this point, this method is constant. It's meant as a starting point for implementing parameters depending on the node i of the Dynkin diagram.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/hecke_algebra_representation.py:docstring of sage.combinat.root_system.hecke_algebra_representation.HeckeAlgebraRepresentation.parameters, line 16.)

Todo

- Non-reduced case (Koornwinder polynomials).
- Non-equal parameters for the affine Hecke algebra.
- Choice of convention (dominant/anti-dominant, ...).
- More uniform implementation of the $T_0^\vee$ operator.
- Optimizations, in particular in the calculation of the eigenvalues for the recursion.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/non_symmetric_macdonald_polynomials.py:docstring of sage.combinat.root_system.non_symmetric_macdonald_polynomials.NonSymmetricMacdonaldPolynomials, line 15.)

Todo

add his notes in latex

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/non_symmetric_macdonald_polynomials.py:docstring of sage.combinat.root_system.non_symmetric_macdonald_polynomials.NonSymmetricMacdonaldPolynomials, line 735.)

Todo

should this just return L in the simply laced case?

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/non_symmetric_macdonald_polynomials.py:docstring of sage.combinat.root_system.non_symmetric_macdonald_polynomials.NonSymmetricMacdonaldPolynomials.L_check, line 3.)

Todo

- Use proposition 6.9 of [Haiman06] to check the action of the Y s on monomials.
- Generalize to any q_1, q_2.
- Check claim by Mark: all scalar products should occur in the finite weight lattice, with alpha 0 being the appropriate projection of the affine alpha 0. Question: can this be emulated by being at level 0?

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/non_symmetric_macdonald_polynomials.py:docstring of sage.combinat.root_system.non_symmetric_macdonald_polynomials.NonSymmetricMacdonaldPolynomials.eigenvalue_experimental, line 87.)

Todo

Could we have nice LaTeX labels in this graph?

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/plot.py:docstring of sage.combinat.root_system.plot, line 452.)

Todo

Display the periodic orientation by adding a $+$ and a $-$ sign close to the label. Typically by using the associated root to shift a bit from the vertex upon which the hyperplane label is attached.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/plot.py:docstring of sage.combinat.root_system.plot.PlotOptions.reflection_hyperplane, line 37.)

Todo

- Optimize the code to only do the embedding/projection for T_0
- Add an option to specify at which level one wants to work. Currently this is level 0.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/root_lattice_realization_algebras.py:docstring of sage.combinat.root_system.root_lattice_realization_algebras.Algebras.ParentMethods.demazure_lusztig_operator_on_classical_on_bas line 12.)

Todo

type free definition (Viviane's definition uses that we are in the ambient space)

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/root_lattice_realization_algebras.py:docstring of sage.combinat.root_system.root_lattice_realization_algebras.Algebras.ParentMethods.divided_difference_on_basis, line 8.)

Todo

make this work for Laurent polynomials too

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/root_lattice_realization_algebras.py:docstring of sage.combinat.root_system.root_lattice_realization_algebras.Algebras.ParentMethods.from_polynomial, line 26.)

Todo

Choose a good set of Cartan Type to run on. Rank >4 is too big. But C_1 and B_1 are boring.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/root_lattice_realization_algebras.py:docstring of sage.combinat.root_system.root_lattice_realization_algebras.Algebras.ParentMethods.twisted_demazure_lusztig_operators, line 75.)

Todo

Investigate why $T_0^\vee$ currently does not satisfy the quadratic relation in type BC. This should hopefuly be fixed when $T_0^\vee$ will have a more uniform implementation:

```
sage: cartan_type = CartanType(["BC",1,2])
sage: KL = RootSystem(cartan_type).weight_lattice().algebra(K)
sage: T = KL.twisted_demazure_lusztig_operators(q1,q2, convention="dominant")
sage: T._test_relations()
... tester.assert_(Ti(Ti(x,i,-q2),i,-q1).is_zero()) ...
AssertionError: False is not true
```

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/root_lattice_realization_algebras.py:docstring of sage.combinat.root_system.root_lattice_realization_algebras.Algebras.ParentMethods.twisted_demazure_lusztig_operators, line 92.)

Todo

This implementation is only valid in the root or weight lattice

---

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/root_lattice_realizations.py:docstring of sage.combinat.root_system.root_lattice_realizations.Root line 7.)

Todo

add a non simply laced example

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/root_lattice_realizations.py:docstring of sage.combinat.root_system.root_lattice_realizations.Root line 17.)

Todo

Provide an option for transparency?

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/root_lattice_realizations.py:docstring of sage.combinat.root_system.root_lattice_realizations.Root line 57.)

Todo

The result should be an enumerated set, and handle infinite root systems.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/root_lattice_realizations.py:docstring of sage.combinat.root_system.root_lattice_realizations.Root line 27.)

Todo

Rename to is_quantum_root

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/root_space.py:docstring of sage.combinat.root_system.root_space.RootSpaceElement.quantum_ro line 13.)

Todo

Lift to CombinatorialFreeModule.Element as canonical_inner_product

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/type_affine.py:docstring of sage.combinat.root_system.type_affine.AmbientSpace.Element.inner_ line 23.)

Todo

Lift to CombinatorialFreeModule.Element as canonical_inner_product

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/type_affine.py:docstring of sage.combinat.root_system.type_affine.AmbientSpace.Element.scalar, line 23.)

Todo

Factor out this code with the classical ambient space.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/type_affine.py:docstring of sage.combinat.root_system.type_affine.AmbientSpace.coroot_lattice, line 6.)

Todo

Factor out this code with the classical ambient space.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/type_affine.py:docstring of sage.combinat.root_system.type_affine.AmbientSpace.simple_coroot, line 16.)

Todo

Currently `subdivide` is currently ignored.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/type_reducible.py:docstring of sage.combinat.root_system.type_reducible.CartanType.cartan_mat line 6.)

Todo

- merge with_apply_multi_module_morphism
- allow for any root space / lattice
- define properly the return type (depends on the base rings of the two spaces)
- make this robust for extended weight lattices (i might be "delta")

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/weight_space.py:docstring of sage.combinat.root_system.weight_space.WeightSpaceElement.scal line 4.)

Todo

Try to compute this directly without actually calculating the full symmetric and exterior squares.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/weyl_characters.py:docstring of sage.combinat.root_system.weyl_characters.WeylCharacterRing. line 17.)

Todo

implement:

- Parabolic subrootsystems
- Parabolic subgroups with a set of nodes as argument

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/root_system/weyl_group.py:docstring of sage.combinat.root_system.weyl_group.ClassicalWeylSubgroup, line 26.)

Todo

delete this class once all coercions will be handled by Sage's coercion model

---

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/sf/classical.py:docstring of sage.combinat.sf.classical.SymmetricFunctionAlgebra_classical, line 3.)

Todo

Is there a not too difficult way to get the power-sum computations to work over any ring, not just one with coercion from $\mathbf{Q}$?

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/sf/monomial.py:docstring of sage.combinat.sf.monomial.SymmetricFunctionAlgebra_monomial.antipode_by_ line 25.)

Todo

Get rid of said technical "reasons".

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/sf/new_kschur.py:docstring of sage.combinat.sf.new_kschur.kSchur, line 93.)

Todo

to be described

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/sf/sf.py:docstring of sage.combinat.sf.sf.SymmetricFunctions, line 644.)

Todo

- Introduce fields with degree 1 elements as in MuPAD-Combinat, to get proper plethysm.
- Use UniqueRepresentation to get rid of all the manual cache handling for the bases
- Devise a mechanism so that pickling bases of symmetric functions pickles the coercions which have a cache.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/sf/sf.py:docstring of sage.combinat.sf.sf.SymmetricFunctions, line 694.)

Todo

Most of the methods in this class are generic (manipulations of morphisms, ...) and should be generalized (or removed)

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/sf/sfa.py:docstring of sage.combinat.sf.sfa.SymmetricFunctionAlgebra_generic, line 3.)

Todo

This method is fast on the monomial and the powersum bases, while all other bases get converted to the monomial basis. For most bases, this is probably the quickest way to do, but at least the Schur basis should have a better option. (Quoting from Stanley's EC2 [STA]: "D. G. Duncan, J. London Math. Soc. 27 (1952), 235-236, or Y. M. Chen, A. M. Garsia, and J. B. Remmel, Contemp. Math. 34 (1984), 109-153".)

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/sf/sfa.py:docstring of sage.combinat.sf.sfa.SymmetricFunctionAlgebra_generic_Element.adams_operation, line 120.)

Todo

This method is fast on the monomial and the powersum bases, while all other bases get converted to the monomial basis. For most bases, this is probably the quickest way to do, but at least the Schur basis should have a better option. (Quoting from Stanley's EC2 [STA]: "D. G. Duncan, J. London Math. Soc. 27 (1952), 235-236, or Y. M. Chen, A. M. Garsia, and J. B. Remmel, Contemp. Math. 34 (1984), 109-153".)

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/sf/sfa.py:docstring of sage.combinat.sf.sfa.SymmetricFunctionAlgebra_generic_Element.frobenius, line 120.)

Todo

This implementation of the reduced Kronecker product is painfully slow.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/sf/sfa.py:docstring of sage.combinat.sf.sfa.SymmetricFunctionAlgebra_generic_Element.reduced_kronecker_p
line 155.)

Todo

This function is an ugly hack using strings. It should be rewritten as soon as the bases of `SymmetricFunctions` are put on a more robust and systematic footing.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/sf/sfa.py:docstring of sage.combinat.sf.sfa.SymmetricFunctionsBases.ParentMethods.corresponding_basis_ove
line 72.)

Todo

generalize to Modules.Graded.Connected.ParentMethods

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/sf/sfa.py:docstring of sage.combinat.sf.sfa.SymmetricFunctionsBases.ParentMethods.one_basis, line 17.)

Todo

As is, this set is essentially the composition of `Compositions(n)` (which give the row lengths) and `SkewPartition(n, row_lengths=...)`, and one would want to "inherit" list and cardinality from this composition.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/skew_partition.py:docstring of sage.combinat.skew_partition.SkewPartitions_n, line 15.)

Todo

- implement subwords with repetitions
- implement the category of EnumeratedMultiset and inheritate from when needed (i.e. the initial word has repeated letters)

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/subword.py:docstring of sage.combinat.subword, line 27.)

Todo

- construct the product of two irreducible representations.
- implement Induction/Restriction of representations.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/symmetric_group_representations.py:docstring of sage.combinat.symmetric_group_representations, line 1.)

Todo

Implement semistandard tableau tuples as defined in [DJM].

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/tableau_tuple.py:docstring of sage.combinat.tableau_tuple, line 180.)

Todo

Add link to some thematic tutorial on graphs

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/tutorial.py:docstring of sage.combinat.tutorial, line 17.)

Todo

add link to some tutorial on quotient rings

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/tutorial.py:docstring of sage.combinat.tutorial, line 468.)

Todo

hide the results by default

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/combinat/tutorial.py:docstring of sage.combinat.tutorial, line 1157.)

Todo

- in case of flood, suggest the user to install the off-line database instead.
- interface with the off-line database (or reimplement it).

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/databases/oeis.py:docstring of sage.databases.oeis, line 123.)

Todo

Ask OEIS for a keyword ensuring that a sequence is infinite.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/databases/oeis.py:docstring of sage.databases.oeis.OEISSequence.is_finite, line 12.)

Todo

- ask OEIS to add a keyword telling whether the sequence comes from a power series, e.g. for http://oeis.org/A000182
- discover other possible conversions.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/databases/oeis.py:docstring of sage.databases.oeis.OEISSequence.natural_object, line 18.)

Todo

ask OEIS to add a "Sage program" field in the database ;)

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/databases/oeis.py:docstring of sage.databases.oeis.OEISSequence.programs, line 12.)

Todo

- in case of flood, suggest the user to install the off-line database instead.
- interface with the off-line database (or reimplement it).

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/databases/oeis.py:docstring of sage.databases.oeis, line 123.)

Todo

Ask OEIS for a keyword ensuring that a sequence is infinite.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/databases/oeis.py:docstring of sage.databases.oeis.OEISSequence.is_finite, line 12.)

Todo

- ask OEIS to add a keyword telling whether the sequence comes from a power series, e.g. for http://oeis.org/A000182
- discover other possible conversions.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/databases/oeis.py:docstring of sage.databases.oeis.OEISSequence.natural_object, line 18.)

Todo

ask OEIS to add a "Sage program" field in the database ;)

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/databases/oeis.py:docstring of sage.databases.oeis.OEISSequence.programs, line 12.)

Todo

- construct as options different string representations for a permutation

- the two intervals: str
- the two intervals on one line: str_one_line
- the separatrix diagram: str_separatrix_diagram
- twin[0] and twin[1] for reduced permutation
- nothing (useful for Rauzy diagram)

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/dynamics/interval_exchanges/template.py:docstring of sage.dynamics.interval_exchanges.template, line 9.)

Todo

This function could probably be made faster.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/rings/algebraic_closure_finite_field.py:docstring of sage.rings.algebraic_closure_finite_field.AlgebraicClosureFiniteFiel
line 12.)

Todo

When trac ticket #10963 is merged we should remove that method and set the category to infinite fields (i.e. `Fields().Infinite()`).

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/rings/algebraic_closure_finite_field.py:docstring of sage.rings.algebraic_closure_finite_field.AlgebraicClosureFiniteFiel
line 5.)

Todo

When trac ticket #10963 is merged we should remove that method and set the category to infinite fields (i.e. `Fields().Infinite()`).

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/rings/algebraic_closure_finite_field.py:docstring of sage.rings.algebraic_closure_finite_field.AlgebraicClosureFiniteFiel
line 3.)

Todo

Implement associated Legendre polynomials and Zernike polynomials. (Neither is in Maxima.) Wikipedia article Associated_Legendre_polynomials Wikipedia article Zernike_polynomials

---

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/functions/orthogonal_polys.py:docstring of sage.functions.orthogonal_polys, line 261.)

Todo

Eventually, category should be `Sets` by default.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/geometry/lattice_polytope.py:docstring of sage.geometry.lattice_polytope.SetOfAllLatticePolytopesClass, line 50.)

Todo

Make it possible to draw Schlegel diagram for 4-polytopes.

```
sage: P=Polyhedron(vertices=[[1,1,0,0],[1,2,0,0],[2,1,0,0],[0,0,1,0],[0,0,0,1]])
sage: P
A 4-dimensional polyhedron in ZZ^4 defined as the convex hull of 5 vertices
sage: P.projection().tikz()
Traceback (most recent call last):
...
NotImplementedError: The polytope has to live in 2 or 3 dimensions.
```

Make it possible to draw 3-polytopes living in higher dimension.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/geometry/polyhedron/plot.py:docstring of sage.geometry.polyhedron.plot.Projection.tikz, line 88.)

Todo

This method sequentially tests each of the forbidden subgraphs in order to know whether the graph is a line graph, which is a very slow method. It could eventually be replaced by `root_graph()` when this method will not require an exponential time to run on general graphs anymore (see its documentation for more information on this problem)... and if it can be improved to return negative certificates !

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/graphs/graph.py:docstring of sage.graphs.graph.Graph.is_line_graph, line 17.)

Todo

Find a beautiful layout for this beautiful graph.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/graphs/graph_generators.py:docstring of sage.graphs.graph_generators.GraphGenerators.SchlaefliGraph, line 14.)

Todo

- Add tooltip like in http://bl.ocks.org/bentwonk/2514276.
- Add a zoom through scrolling (http://bl.ocks.org/mbostock/3681006).

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/graphs/graph_plot_js.py:docstring of sage.graphs.graph_plot_js, line 54.)

Todo

Technical things:

- Query the database for non-inclusion results so that comparisons can return `False`, and implement strict inclusions.
- Implement a proper search method for the classes not listed in `graph_classes`
- Some of the graph classes appearing in `graph_classes` already have a recognition algorithm implemented in Sage. It would be so nice to be able to write `g in Trees`, `g in Perfect`, `g in Chordal`, ... :-)

Long-term stuff:

- Implement simple accessors for all the information in the ISGCI database (as can be done from the website)
- Implement intersection of graph classes
- Write generic recognition algorithms for specific classes (when a graph class is defined by the exclusion of subgraphs, one can write a generic algorithm checking the existence of each of the graphs, and this method already exists in Sage).
- Improve the performance of Sage's graph library by letting it take advantage of the properties of graph classes. For example, `Graph.independent_set()` could use the library to detect that a given graph is, say, a tree or a planar graph, and use a specialized algorithm for finding an independent set.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/graphs/isgci.py:docstring of sage.graphs.isgci, line 282.)

Todo

This method sequentially tests each of the forbidden subgraphs in order to know whether the graph is a line graph, which is a very slow method. It could eventually be replaced by `root_graph()` when this method will not require an exponential time to run on general graphs anymore (see its documentation for more information on this problem)... and if it can be improved to return negative certificates !

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/graphs/line_graph.py:docstring of sage.graphs.line_graph.is_line_graph, line 17.)

Todo

This code could probably be made more efficient by using FLINT polynomials and being written in Cython, using an array of fmpz_poly_t pointers or something... Right now just about the whole complement optimization is written in Python, and could be easily sped up.

(The *original entry* is located in docstring of sage.graphs.matchpoly.complete_poly, line 7.)

Todo

- Implement a non-naive fallback method for computing all the elements of the conjugacy class when the group is not defined in GAP, as the one in Butler's paper.
- Define a sage method for gap matrices so that groups of matrices can use the quicker GAP algorithm rather than the naive one.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/groups/conjugacy_classes.py:docstring of sage.groups.conjugacy_classes, line 9.)

Todo

Implement a non-naive algorithm, cf. for instance G. Butler: "An Inductive Schema for Computing Conjugacy Classes in Permutation Groups", Math. of Comp. Vol. 62, No. 205 (1994)

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/groups/conjugacy_classes.py:docstring of sage.groups.conjugacy_classes.ConjugacyClass.set, line 4.)

Todo

- Include support for different orderings (currently only shortlex is used).
- Include the GAP package kbmag for more functionalities, including automatic structures and faster compiled functions.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/groups/finitely_presented.py:docstring of sage.groups.finitely_presented.RewritingSystem, line 46.)

Todo

Currently the label ∞ is implemented as -1 in the Coxeter matrix.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/groups/matrix_gps/coxeter_group.py:docstring of sage.groups.matrix_gps.coxeter_group.CoxeterMatrixGroup, line 31.)

Todo

Fix the broken hash.

```
sage: G = SymmetricGroup(6)
sage: G3 = G.subgroup([G((1,2,3,4,5,6)),G((1,2))])
sage: hash(G) == hash(G3)  # todo: Should be True!
False
```

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/groups/perm_gps/permgroup_named.py:docstring of sage.groups.perm_gps.permgroup_named.PermutationGroup_uniqu line 1.)

Todo

Up to now, this group is only implemented for finite fields because of the limited support of automorphisms for arbitrary rings.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/groups/semimonomial_transformations/semimonomial_transformation_group.py:docstring of sage.groups.semimonomial_transformations.semimonomial_transformation_group, line 29.)

Todo

Up to now, this group is only implemented for finite fields because of the limited support of automorphisms for arbitrary rings.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/groups/semimonomial_transformations/semimonomial_transformation_group.py:docstring of sage.groups.semimonomial_transformations.semimonomial_transformation_group.SemimonomialTransformationGroup, line 31.)

Todo

Create an animated image file (GIF) if spin is on and put data extracted from a file into a variable/string/structure to return

---

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/interfaces/jmoldata.py:docstring of sage.interfaces.jmoldata.JmolData, line 1.)

Todo

use this library in the `SymmetricFunctions` code, to make it easy to apply it to linear combinations of Schur functions.

(The *original entry* is located in docstring of sage.libs.lrcalc.lrcalc, line 76.)

Todo

Implement this method.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/logic/logic.py:docstring of sage.logic.logic.SymbolicLogic.prove, line 4.)

Todo

Implement this method.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/logic/logic.py:docstring of sage.logic.logic.SymbolicLogic.simplify, line 4.)

Todo

Implement faster algorithms, including a division-free one. Does [Rote2001], section 3.3 give one?

Check the implementation of the matchings used here for performance?

(The *original entry* is located in docstring of sage.matrix.matrix2.Matrix.pfaffian, line 82.)

Todo

Write abstract `RelabeledMatroid` class, and add `relabel()` method to the main `Matroid` class, together with `_relabel()` method that can be replaced by subclasses. Use the code from `is_isomorphism()` in `relabel()` to deal with a variety of input methods for the relabeling.

(The *original entry* is located in docstring of sage.matroids.basis_matroid.BasisMatroid.relabel, line 15.)

Todo

Add optional argument `groundset` to each method so users can customize the groundset of the matroid. We probably want some means of relabeling to accomplish that.

Add option to specify the field for represented matroids.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/matroids/catalog.py:docstring of sage.matroids.catalog, line 10.)

Todo

Optional arguments `ring` and `x`, such that the resulting matroid is represented over `ring` by a reduced matrix like `[-1 0 x] [ 1 -1 0] [ 0 1 -1]`

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/matroids/catalog.py:docstring of sage.matroids.catalog.Whirl, line 28.)

Todo

This important method can (and should) be optimized considerably. See [Hlineny] p.1219 for hints to that end.

(The *original entry* is located in docstring of sage.matroids.linear_matroid.LinearMatroid.has_field_minor, line 16.)

Todo

This important method can (and should) be optimized considerably. See [Hlineny] p.1219 for hints to that end.

(The *original entry* is located in docstring of sage.matroids.matroid.Matroid.has_minor, line 16.)

Todo

Implement this using the efficient algorithm from [BC79].

(The *original entry* is located in docstring of sage.matroids.matroid.Matroid.is_3connected, line 11.)

Todo

Make implementation more efficient, e.g. generalizing the approach from trac ticket #1314 from graphs to matroids.

(The *original entry* is located in docstring of sage.matroids.matroid.Matroid.tutte_polynomial, line 31.)

Todo

- Handle nicely (covariant) functorial constructions and axioms

---

(The *original entry* is located in docstring of sage.misc.c3_controlled.CmpKey, line 45.)

Todo

Make the following work nicely:

```
sage: b.x?                # todo: not implemented
sage: b.x??               # todo: not implemented
```

(The *original entry* is located in docstring of sage.misc.lazy_attribute.lazy_attribute, line 175.)

Todo

Improve the error message:

```
sage: B().unimplemented_A # todo: not implemented
Traceback (most recent call last):
...
AttributeError: 'super' object has no attribute 'unimplemented_A'
```

(The *original entry* is located in docstring of sage.misc.lazy_attribute.lazy_attribute, line 387.)

Todo

- Add Pyrex source code inspection (I assume it doesn't currently do this)
- Add ability to sort output by time
- Add option to constructor to print timing immediately when checkpoint is reached
- Migrate to Pyrex?
- Add ability to return timings in a more machine-friendly format

---

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/misc/profiler.py:docstring of sage.misc.profiler.Profiler, line 40.)

Todo

This should be moved to `sage.matrix.matrix_modn_dense` at some point.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/modular/overconvergent/hecke_series.py:docstring of sage.modular.overconvergent.hecke_series.ech_form, line 4.)

Todo

Refactor modules such that it only counts what category the base ring belongs to, but not what is its Python class.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/modules/free_module.py:docstring of sage.modules.free_module.FreeModuleFactory, line 122.)

Todo

Should implement a version of the algorithm that guarantees correct output. See Algorithm 4 in [Doyle-Krumm] for details of an implementation that takes precision issues into account.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/rings/number_field/number_field.py:docstring of sage.rings.number_field.number_field.NumberField_absolute.elements_
line 34.)

Todo

The `_new()` method should be overridden in this class to copy the `D` and `standard_embedding` attributes

---

(The *original entry* is located in docstring of sage.rings.number_field.number_field_element_quadratic, line 14.)

Todo

doctests for converting from other types of p-adic rings

(The *original entry* is located in docstring of sage.rings.padics.padic_fixed_mod_element.pAdicFixedModElement, line 82.)

Todo

(The *original entry* is located in docstring of sage.rings.padics.padic_generic_element.pAdicGenericElement.log, line 46.)

Todo

See comments at trac ticket #4805. Currently the absolute precision of the result may be less than the given value of absprec, and error-handling is imperfect.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/schemes/elliptic_curves/ell_point.py:docstring of sage.schemes.elliptic_curves.ell_point.EllipticCurvePoint_number_fiel line 26.)

Todo

make `GaloisAutomorphism` derive from GroupElement, so that one gets powers for free, etc.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/schemes/elliptic_curves/heegner.py:docstring of sage.schemes.elliptic_curves.heegner.GaloisAutomorphism, line 3.)

Todo

Unify `UnionOfIntervals` with the class `RealSet` introduced by trac ticket #13125; see trac ticket #16063.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/schemes/elliptic_curves/height.py:docstring of sage.schemes.elliptic_curves.height.UnionOfIntervals, line 19.)

Todo

Eventually we will want to run this in characteristic 3, so we need to: (a) Allow $Q(x)$ to contain an x^2 term, and (b) Remove the requirement that 3 be invertible. Currently this is used in the Toom-Cook algorithm to speed multiplication.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/schemes/hyperelliptic_curves/monsky_washnitzer.py:docstring of sage.schemes.hyperelliptic_curves.monsky_washnitze line 17.)

Todo

write an example checking multiplication of these polynomials against Sage's ordinary quotient ring arithmetic. I can't seem to get the quotient ring stuff happening right now...

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/schemes/hyperelliptic_curves/monsky_washnitzer.py:docstring of sage.schemes.hyperelliptic_curves.monsky_washnitze line 66.)

Todo

Working with sparse matrices should usually give faster results, but with the current implementation it actually works slower. There should be a way to improve performance with regards to this.

(The *original entry* is located in docstring of sage.rings.polynomial.multi_polynomial_ring_generic.MPolynomialRing_generic.macaula line 39.)

Todo

What should we do about this method? Is nilpotency of a power series even decidable (assuming a nilpotency oracle in the base ring)? And I am not sure that returning `True` just because the series has finite precision and zero constant term is a good idea.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/rings/multi_power_series_ring_element.py:docstring of sage.rings.multi_power_series_ring_element.MPowerSeries.is_n line 13.)

Todo

This currently does not work for quivers with cycles, even if there are only finitely many paths from `start` to `end`.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/quivers/path_semigroup.py:docstring of sage.quivers.path_semigroup.PathSemigroup.all_paths, line 16.)

Todo

Change the wording `Reverse of ()` into something more meaningful.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/quivers/representation.py:docstring of sage.quivers.representation, line 389.)

Todo

Implement this method.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/rings/ideal.py:docstring of sage.rings.ideal.Ideal_generic.absolute_norm, line 9.)

Todo

This is not implemented for many rings. Implement it!

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/rings/ideal.py:docstring of sage.rings.ideal.Ideal_generic.is_maximal, line 4.)

Todo

Code is naive. Only keeps track of ideal generators as set during initialization of the ideal. (Can the base ring change? See example below.)

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/rings/ideal.py:docstring of sage.rings.ideal.Ideal_generic.is_principal, line 4.)

Todo

The following skipped tests should be removed once trac ticket #13999 is fixed:

```
sage: TestSuite(S).run(skip=['_test_nonzero_equal', '_test_elements', '_test_zero'])
```

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/rings/quotient_ring.py:docstring of sage.rings.quotient_ring, line 15.)

Todo

Not yet implemented!

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/rings/quotient_ring.py:docstring of sage.rings.quotient_ring.QuotientRing_nc.characteristic, line 3.)

Todo

Note that `ngens` counts 0 as a generator. Does this make sense? That is, since 0 only generates itself and the fact that this is true for all rings, is there a way to "knock it off" of the generators list if a generator of some original ring is modded out?

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/rings/quotient_ring.py:docstring of sage.rings.quotient_ring.QuotientRing_nc.ngens, line 3.)

Todo

Implement `ComplexIntervalFieldElement` multiplicative order similar to `ComplexNumber` multiplicative order with `_set_multiplicative_order(n)` and `ComplexNumber.multiplicative_order()` methods.

(The *original entry* is located in docstring of sage.rings.complex_interval, line 23.)

Todo

Implement `ComplexIntervalFieldElement` multiplicative order and set this output to have multiplicative order `n`.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/rings/complex_interval_field.py:docstring of sage.rings.complex_interval_field.ComplexIntervalField_class.zeta, line 3.)

Todo

- implementation of matrices over the universal cyclotomic field.

- speed improvements of the cythonized methods.
- speed improvements for scalar multiples.
- Remove the inheritance from Field and FieldElement as soon as the methods `is_field(proof=True)` is implemented in the Fields category.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/rings/universal_cyclotomic_field/universal_cyclotomic_field.py:docstring of sage.rings.universal_cyclotomic_field.universal_cyclotomic_field, line 23.)

Todo

add heights to integer.pyx and remove special case

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/schemes/affine/affine_morphism.py:docstring of sage.schemes.affine.affine_morphism.SchemeMorphism_polynomial_af line 30.)

Todo

This could be improved.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/schemes/affine/affine_morphism.py:docstring of sage.schemes.affine.affine_morphism.SchemeMorphism_polynomial_af line 7.)

Todo

p-adic heights

add heights to integer.pyx and remove special case

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/schemes/affine/affine_point.py:docstring of sage.schemes.affine.affine_point.SchemeMorphism_point_affine.global_heig line 34.)

Todo

Currently, `SchemeMorphism` copies code from `Map` rather than inheriting from it. This is to work around a bug in Cython: We want to create a common sub-class of `ModuleElement` and `SchemeMorphism`, but Cython would currently confuse cpdef attributes of the two base classes. Proper inheritance should be used as soon as this bug is fixed.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/schemes/generic/morphism.py:docstring of sage.schemes.generic.morphism.SchemeMorphism, line 7.)

Todo

Do the division when the base ring is p-adic or a function field so that the output is a polynomial.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/schemes/projective/projective_morphism.py:docstring of sage.schemes.projective.projective_morphism.SchemeMorphis line 41.)

Todo

It would be nice to get this to actually be a polynomial.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/schemes/projective/projective_morphism.py:docstring of sage.schemes.projective.projective_morphism.SchemeMorphis line 124.)

Todo

add heights to integer.pyx and remove special case

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/schemes/projective/projective_morphism.py:docstring of sage.schemes.projective.projective_morphism.SchemeMorphis line 42.)

Todo

would be better to keep the dehomogenizations for reuse

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/schemes/projective/projective_morphism.py:docstring of sage.schemes.projective.projective_morphism.SchemeMorphis line 70.)

Todo

Is there a more efficient way to do this?

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/schemes/projective/projective_morphism.py:docstring of sage.schemes.projective.projective_morphism.SchemeMorphis line 7.)

Todo

This could be improved.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/schemes/projective/projective_morphism.py:docstring of sage.schemes.projective.projective_morphism.SchemeMorphis line 9.)

Todo

- move some of this to Cython so that it is faster especially the possible periods mod p.
- have the last prime of good redution used also return the list of points instead of getting the information again for all_points.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/schemes/projective/projective_morphism.py:docstring of sage.schemes.projective.projective_morphism.SchemeMorphis line 66.)

Todo

- do not return duplicate points
- improve hash to reduce memory of pointtable

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/schemes/projective/projective_morphism.py:docstring of sage.schemes.projective.projective_morphism.SchemeMorphis line 41.)

Todo

p-adic heights

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/schemes/projective/projective_point.py:docstring of sage.schemes.projective.projective_point.SchemeMorphism_point_p line 34.)

Todo

error bounds for dimension > 1

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/schemes/projective/projective_point.py:docstring of sage.schemes.projective.projective_point.SchemeMorphism_point_p line 46.)

Todo

Is there a more efficient way to do this?

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/schemes/projective/projective_point.py:docstring of sage.schemes.projective.projective_point.SchemeMorphism_point_p line 52.)

Todo

good name?

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/sets/family.py:docstring of sage.sets.family.AbstractFamily.map, line 4.)

Todo

generalize to any number of families and merge with map?

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/sets/family.py:docstring of sage.sets.family.AbstractFamily.zip, line 5.)

Todo

FIXME: What should be the order of the result? That of `self.object()`? Or the order given by `set(self.object())`? Note that `__getitem__()` is currently implemented in term of this list method, which is really inefficient ...

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/sets/set.py:docstring of sage.sets.set.Set_object_enumerated.list, line 13.)

Todo

It is not yet possible to use `set_from_method` in conjunction with `cached_method`.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/sets/set_from_iterator.py:docstring of sage.sets.set_from_iterator.EnumeratedSetFromIterator_method_decorator, line 66.)

Todo

Move this method elsewhere (typically in the Modules category) so as not to pollute the namespace of all category objects.

(The *original entry* is located in docstring of sage.structure.category_object.CategoryObject.base_ring, line 33.)

Todo

title

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/structure/global_options.py:docstring of sage.structure.global_options.GlobalOptions.dispatch, line 1.)

Todo

Eventually, category should be `Sets` by default.

(The *original entry* is located in docstring of sage.structure.parent.Parent, line 50.)

Todo

Create a custom-made `SourPickle` for the last example.

(The *original entry* is located in docstring of sage.structure.sage_object.unpickle_all, line 62.)

Todo

Illustrate how this can be fixed on a case by case basis.

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/structure/unique_representation.py:docstring of sage.structure.unique_representation.CachedRepresentation, line 361.)

Todo

should reuse something preexisting ...

(The *original entry* is located in /home/sagemath/build-sage/sage-6.3/local/lib/python2.7/site-packages/sage/structure/unique_representation.py:docstring of sage.structure.unique_representation.unreduce, line 6.)

- `History and License`

1.11 Indices and Tables

- *genindex*
- *modindex*
- *search*