Infrastructure for generic code in SageMath : categories, axioms, constructions

Nicolas M. Thiéry

LRI, Université Paris Sud 11

May 31th of 2016, Sage Days 74, Meudon

SageMath : a general purpose software for mathematics

Numbers 42, $\frac{7}{9}$, $\frac{I+sqrt(3)}{2}$, π, 2.71828182845904523536028747?

Matrices : $\begin{pmatrix} 4 & -1 & 1 & -1 \\ -1 & 2 & -1 & -1 \\ 0 & 5 & 1 & 3 \end{pmatrix}$

Polynomials : $-9x^8 + x^7 + x^6 - 13x^5 - x^3 - 3x^2 - 8x + 4$

Series : $1 + 1x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \frac{1}{120}x^5 + \cdots$

Finite fields, algebraic extensions, elliptic curves, ...

Symbolic expressions, equations, ...

SageMath : a general purpose software for mathematics

Numbers 42, $\frac{7}{9}$, $\frac{I+sqrt(3)}{2}$, π, 2.71828182845904523536028747?

Matrices : $\begin{pmatrix} 4 & -1 & 1 & -1 \\ -1 & 2 & -1 & -1 \\ 0 & 5 & 1 & 3 \end{pmatrix}$

Polynomials : $-9x^8 + x^7 + x^6 - 13x^5 - x^3 - 3x^2 - 8x + 4$

Series : $1 + 1x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \frac{1}{120}x^5 + \cdots$

Finite fields, algebraic extensions, elliptic curves, ...

Symbolic expressions, equations, ...

SageMath : a general purpose software for mathematics

Numbers 42, $\frac{7}{9}$, $\frac{I+sqrt(3)}{2}$, π, 2.71828182845904523536028747?

Matrices : $\begin{pmatrix} 4 & -1 & 1 & -1 \\ -1 & 2 & -1 & -1 \\ 0 & 5 & 1 & 3 \end{pmatrix}$

Polynomials : $-9x^8 + x^7 + x^6 - 13x^5 - x^3 - 3x^2 - 8x + 4$

Series : $1 + 1x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \frac{1}{120}x^5 + \cdots$

Finite fields, algebraic extensions, elliptic curves, ...

Symbolic expressions, equations, ...

SageMath : a general purpose software for mathematics

Numbers 42, $\frac{7}{9}$, $\frac{I+sqrt(3)}{2}$, π, 2.718281828459045235360028747?

Matrices : $\begin{pmatrix} 4 & -1 & 1 & -1 \\ -1 & 2 & -1 & -1 \\ 0 & 5 & 1 & 3 \end{pmatrix}$

Polynomials : $-9x^8 + x^7 + x^6 - 13x^5 - x^3 - 3x^2 - 8x + 4$

Series : $1 + 1x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \frac{1}{120}x^5 + \cdots$

Finite fields, algebraic extensions, elliptic curves, ...

Symbolic expressions, equations, ...

SageMath : a general purpose software for mathematics

Numbers 42, $\frac{7}{9}$, $\frac{I+sqrt(3)}{2}$, π, 2.71828182845904523536028747?

Matrices : $\begin{pmatrix} 4 & -1 & 1 & -1 \\ -1 & 2 & -1 & -1 \\ 0 & 5 & 1 & 3 \end{pmatrix}$

Polynomials : $-9x^8 + x^7 + x^6 - 13x^5 - x^3 - 3x^2 - 8x + 4$

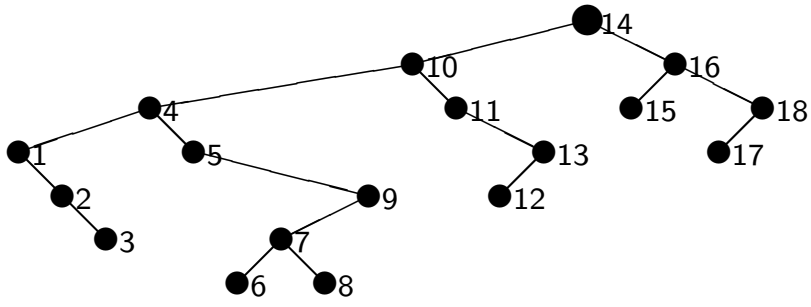
Series : $1 + 1x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \frac{1}{120}x^5 + \cdots$

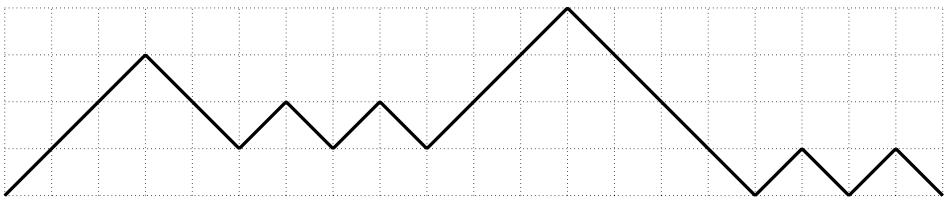
Finite fields, algebraic extensions, elliptic curves, ...

Symbolic expressions, equations, ...

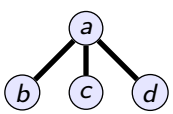
Combinatorial objects

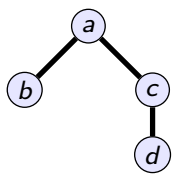
1	3	4	7
2	5	6	
8			

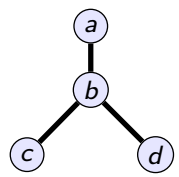




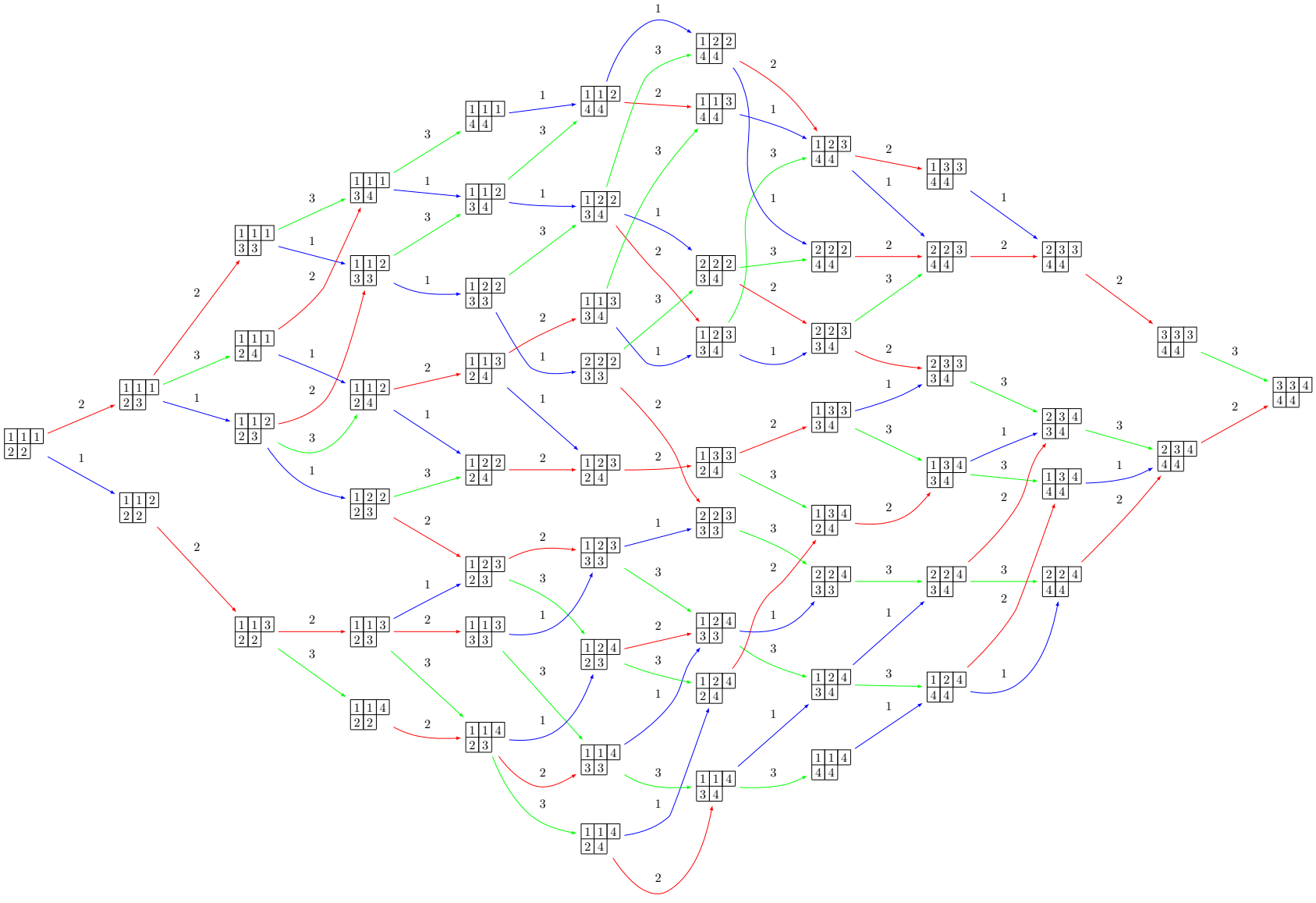
$$0100101001001010010100100101001001010010\cdots$$

$$\frac{\frac{1}{6}q^2-\frac{1}{6}q}{q^5+2q^4+3q^3+3q^2+2q+1}$$ 

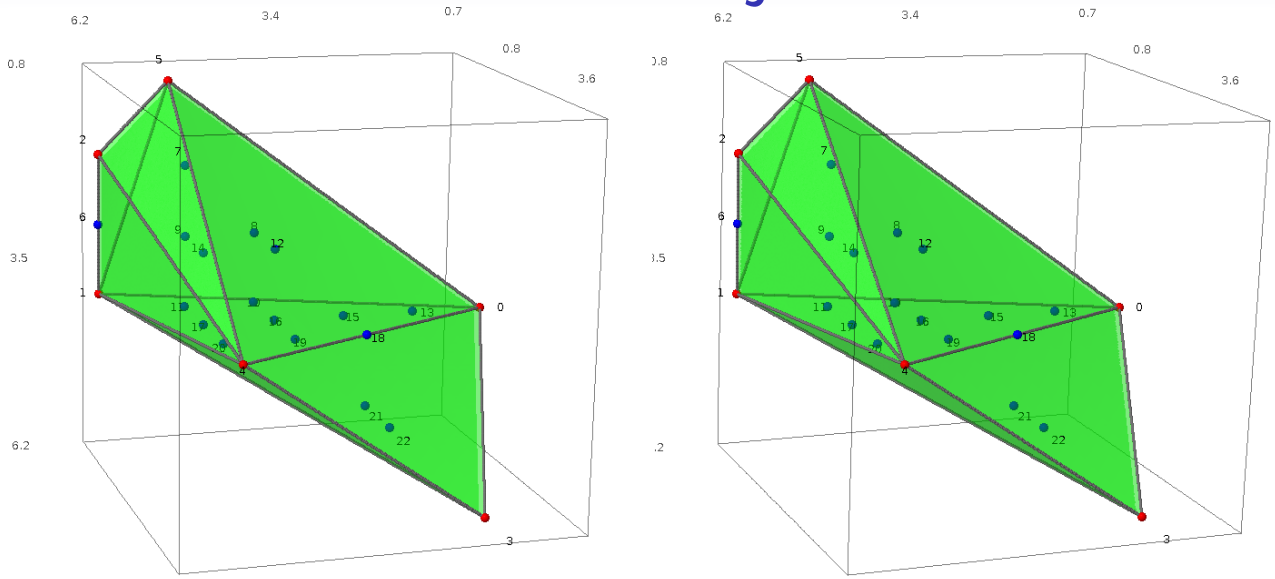
$$+\frac{q^2}{q^5+2q^4+3q^3+3q^2+2q+1}$$ 

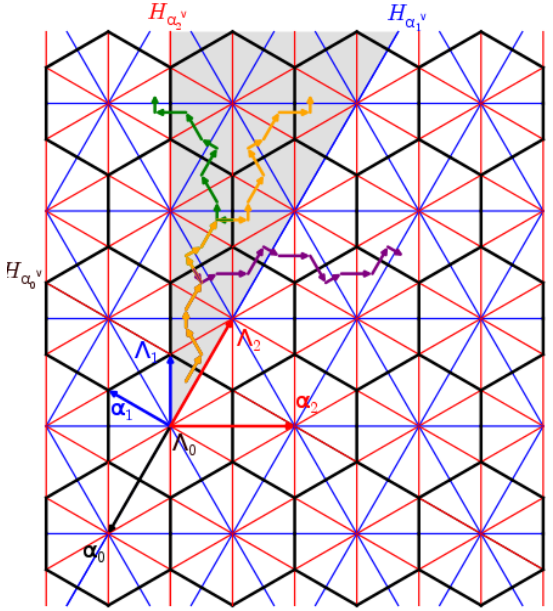
$$+\frac{\frac{1}{2}q}{q^4+q^3+2q^2+q+1}$$ 

Graphs



Geometric objects





Sage : a large library of mathematical objects and algorithms

- 1.5M lines of code/doc/tests (Python/Cython) plus dependencies
- 1k+ types of objets
- 2k+ methods and functions
- 200 regular contributors

Problems

- How to structure this library
- How to guide the user
- How to promote consistency and robustness ?
- How to reduce duplication ?

Example : binary powering

```
sage : m = 3
sage : m^8 == m*m*m*m*m*m*m*m == ((m^2)^2)^2
True
```

```
sage : m = random_matrix(QQ, 4)
sage : m^8 == m*m*m*m*m*m*m*m == ((m^2)^2)^2
True
```

- Complexity : $O(\log(k))$ instead of $O(k)$!
- We would want a single *generic* implementation !

Example : binary powering II

Algebraic context

- *Semigroup* : a set S endowed with an associative binary internal law $*$
- The integers form a semigroup
- Square matrices form a semigroup

We want

- Implement pow_exp(x,k)
- Specify that
  - if x is an *element* of a semigroup
  - then x^k can be computed with pow_exp(x,k)

What happens if :

- x is an element of a group ?
- x is an element of a ring in small caracteristic ?

Example : binary powering II

Algebraic context

- *Semigroup* : a set S endowed with an associative binary internal law $*$
- The integers form a semigroup
- Square matrices form a semigroup

We want

- Implement pow_exp(x,k)
- Specify that
  - if x is an *element* of a semigroup
  - then x^k can be computed with pow_exp(x,k)

What happens if :

- x is an element of a group ?
- x is an element of a ring in small caracteristic ?

Example : binary powering II

Algebraic context

- *Semigroup* : a set S endowed with an associative binary internal law $*$
- The integers form a semigroup
- Square matrices form a semigroup

We want

- Implement pow_exp(x,k)
- Specify that
  - if x is an *element* of a semigroup
  - then x^k can be computed with pow_exp(x,k)

What happens if :

- x is an element of a group ?
- x is an element of a ring in small caracteristic ?

Selection mechanism

We want

- Design a hierarchy of contexts and specify the operations there
- Provide generic implementations of those operations
- Specify in which context they are valid
- Specify in which context each object is

We need a *selection mechanism* :
to resolve the call `f(x)` by selecting the most specific implementation of the operation `f`

Selection mechanism

We want

- Design a hierarchy of contexts and specify the operations there
- Provide generic implementations of those operations
- Specify in which context they are valid
- Specify in which context each object is

We need a *selection mechanism* :

to resolve the call `f(x)` by selecting the most specific implementation of the operation `f`

Hierarchy of contexts for mathematics

In general

Hard problem : isolate the proper business concepts

In mathematics

- A small number of fundamental concepts : operations, axioms
- Concepts known by the users
- The richness comes from *combining* those few concepts to form many contexts
  Field : +, *, associative, commutative, distributive, ...

Hierarchy of contexts for mathematics

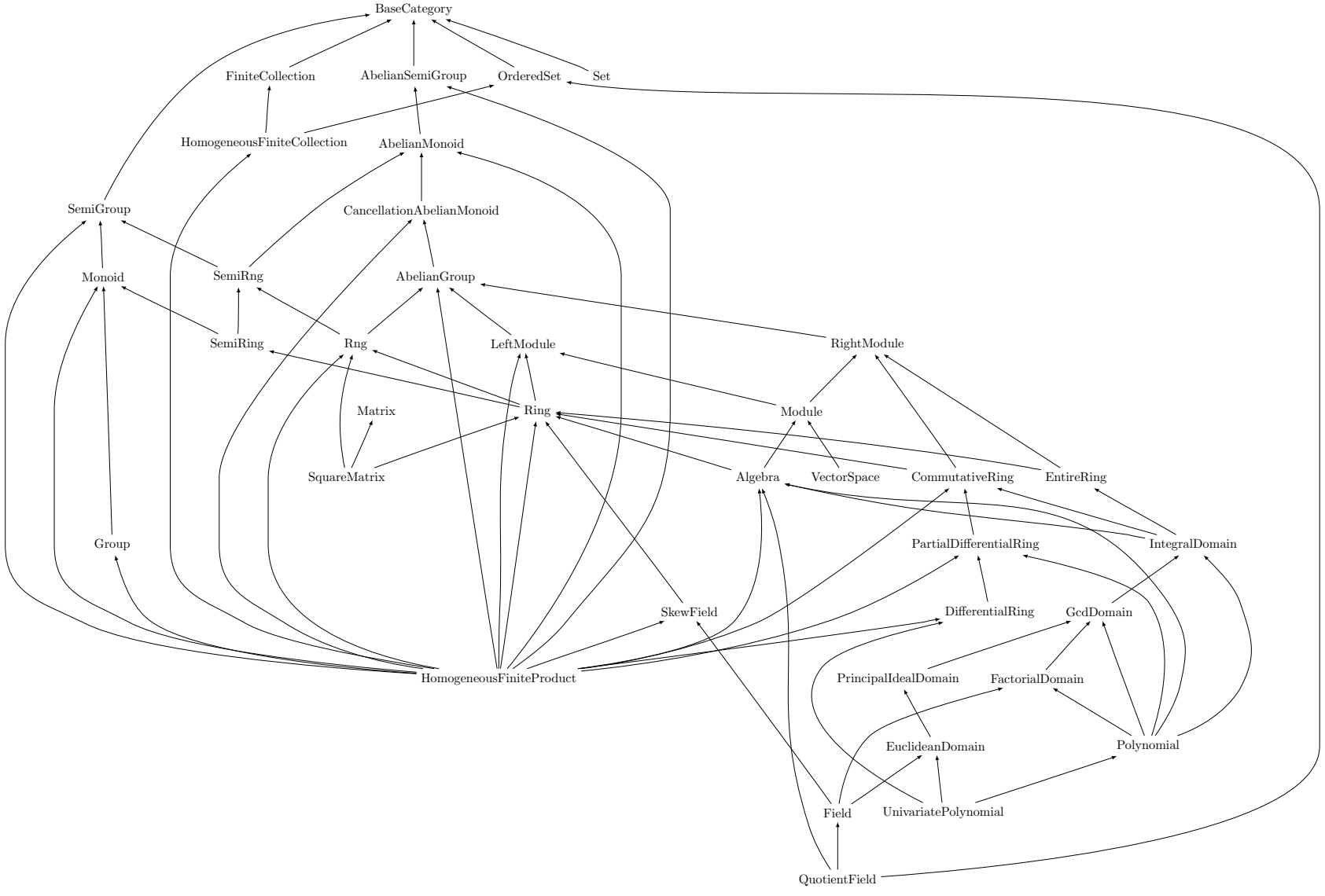
In general

Hard problem : isolate the proper business concepts

In mathematics

- A small number of fundamental concepts :
  operations, axioms
- Concepts known by the users
- The richness comes from *combining* those few concepts to form many contexts
  Field : $+$, *, associative, commutative, distributive, ...

A hierarchy of contexts based on mathematical categories



A robust hierarchy based on a century of abstract algebra

Pioneers 1980- I

Axiom, Aldor, MuPAD

-
- Selection mechanism : roughly object oriented programming
- Hierarchy of abstract classes modeling the mathematical categories

Exemple

```
category Semigroups :
    category Magmas;

    intpow := proc(x, k) ...
    // other methods
```

Pioneers 1980- I

Axiom, Aldor, MuPAD

-
- Selection mechanism : roughly object oriented programming
- Hierarchy of abstract classes modeling the mathematical categories

Exemple

```
category Semigroups :
    category Magmas;

    intpow := proc(x, k) ...
    // other methods
```

Pioneers 1980- I

Axiom, Aldor, MuPAD

-
- Selection mechanism : roughly object oriented programming
- Hierarchy of abstract classes modeling the mathematical categories

Exemple

```
category Semigroups :
    category Magmas;

    intpow := proc(x, k) ...
    // other methods
```

Pioneers 1980- II

GAP

- Specific language
- One filter per basic concept : `IsMagma(G)`, `IsAssociative(G)`, ...
- `InstallMethod(Operation, filters, method)`
- Method selection according to the filters that are know to be satisfied by x
- Implicit modeling of the hierarchy

Exemple

```
powExp := function(n, k) ...

InstallMethod(pow, [IsMagma, IsAssociative],
              powExp)
```

Pioneers 1980- II

GAP

- Specific language
- One filter per basic concept : `IsMagma(G)`, `IsAssociative(G)`, ...
- `InstallMethod(Operation, filters, method)`
- Method selection according to the filters that are know to be satisfied by x
- Implicit modeling of the hierarchy

Exemple

```
powExp := function(n, k) ...

InstallMethod(pow, [IsMagma, IsAssociative],
              powExp)
```

Pioneers 1980- II

GAP

- Specific language
- One filter per basic concept : `IsMagma(G)`, `IsAssociative(G)`, ...
- `InstallMethod(Operation, filters, method)`
- Method selection according to the filters that are know to be satisfied by x
- Implicit modeling of the hierarchy

Exemple

```
powExp := function(n, k) ...

InstallMethod(pow, [IsMagma, IsAssociative],
                   powExp)
```

Implementation in Sage (2008-)

Strategical choices

- A standard language (Python)
- Selection mechanism : object oriented programming

Constraints

- Partial compilation (Cython), serialization
- Multiple inheritance with Python / Cython
- Scaling !

Spécificités

- Distinction Élément/Parent (à la Magma)
- Morphismes
- Constructions fonctorielles
- Axiomes

Implementation in Sage (2008-)

Strategical choices

- A standard language (Python)
- Selection mechanism : object oriented programming

Constraints

- Partial compilation (Cython), serialization
- Multiple inheritance with Python / Cython
- Scaling !

Spécificités

- Distinction Élément/Parent (à la Magma)
- Morphismes
- Constructions fonctorielles
- Axiomes

Implementation in Sage (2008-)

Strategical choices

- A standard language (Python)
- Selection mechanism : object oriented programming

Constraints

- Partial compilation (Cython), serialization
- Multiple inheritance with Python / Cython
- Scaling !

Spécificités

- Distinction Élément/Parent (à la Magma)
- Morphismes
- Constructions fonctorielles
- Axiomes

Implementation in Sage (2008-)

Strategical choices

- A standard language (Python)
- Selection mechanism : object oriented programming

Constraints

- Partial compilation (Cython), serialization
- Multiple inheritance with Python / Cython
- Scaling !

Spécificités

- Distinction Élément/Parent (à la Magma)
- Morphismes
- Constructions fonctorielles
- Axiomes

In practice

Let's write a parent and a category.

Implementation in Sage

Parents, Elements, Morphisms

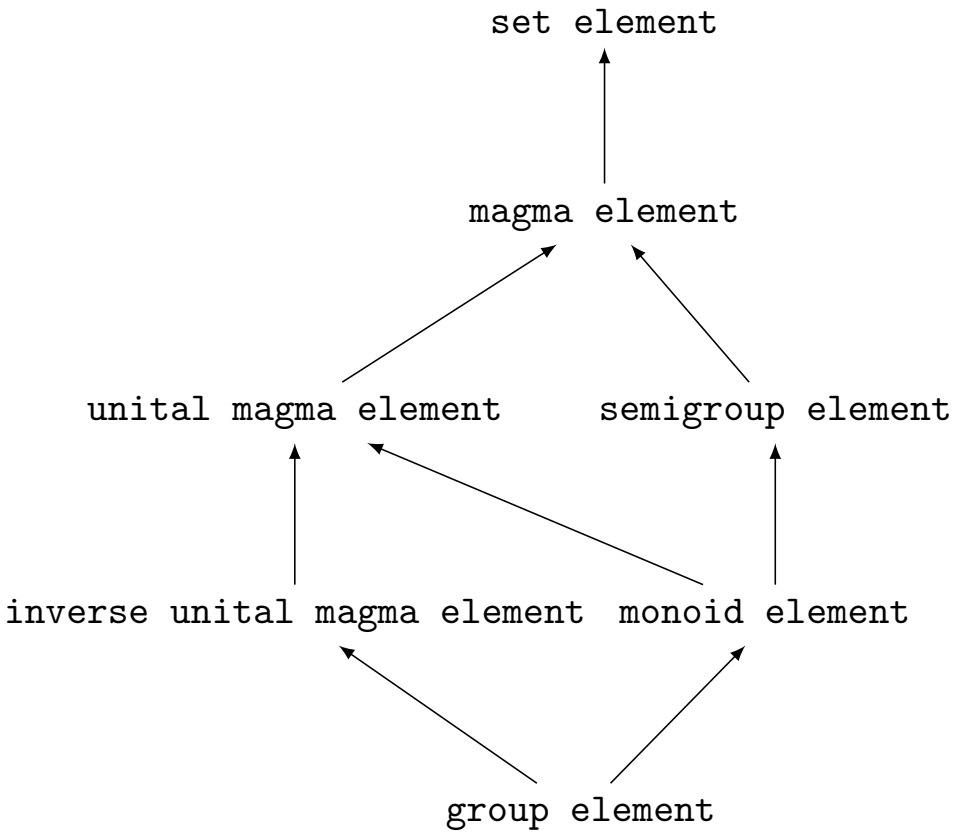
Category

A *bookshelf* about a given context :

- Semantic information
- Mixins for parents, elements, morphisms :
  - Documentation
  - Generic code
  - Tests

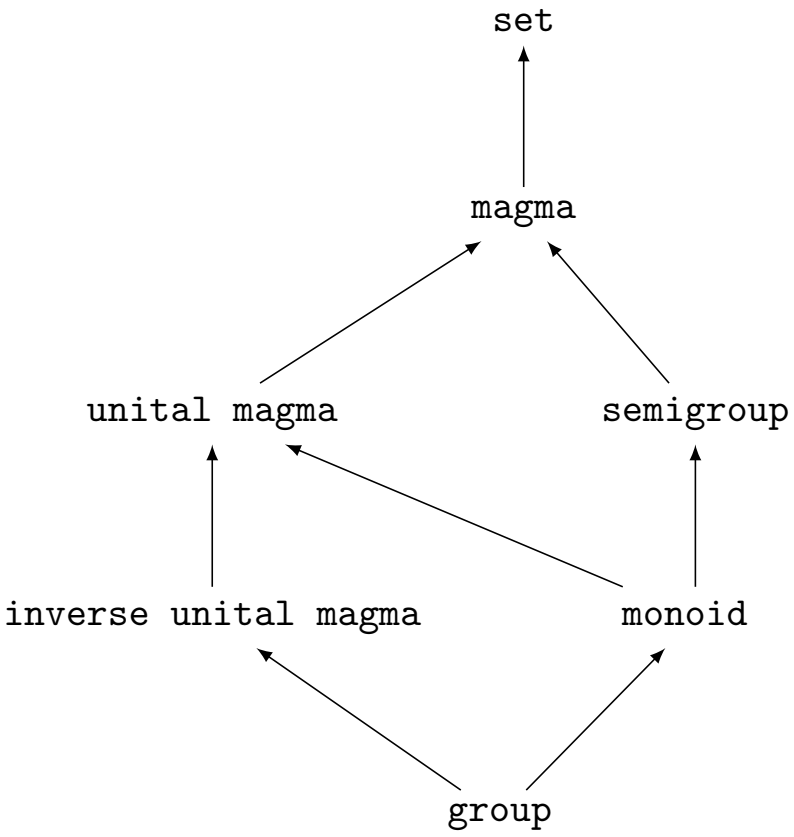
Parent, éléments, morphismes, catégories

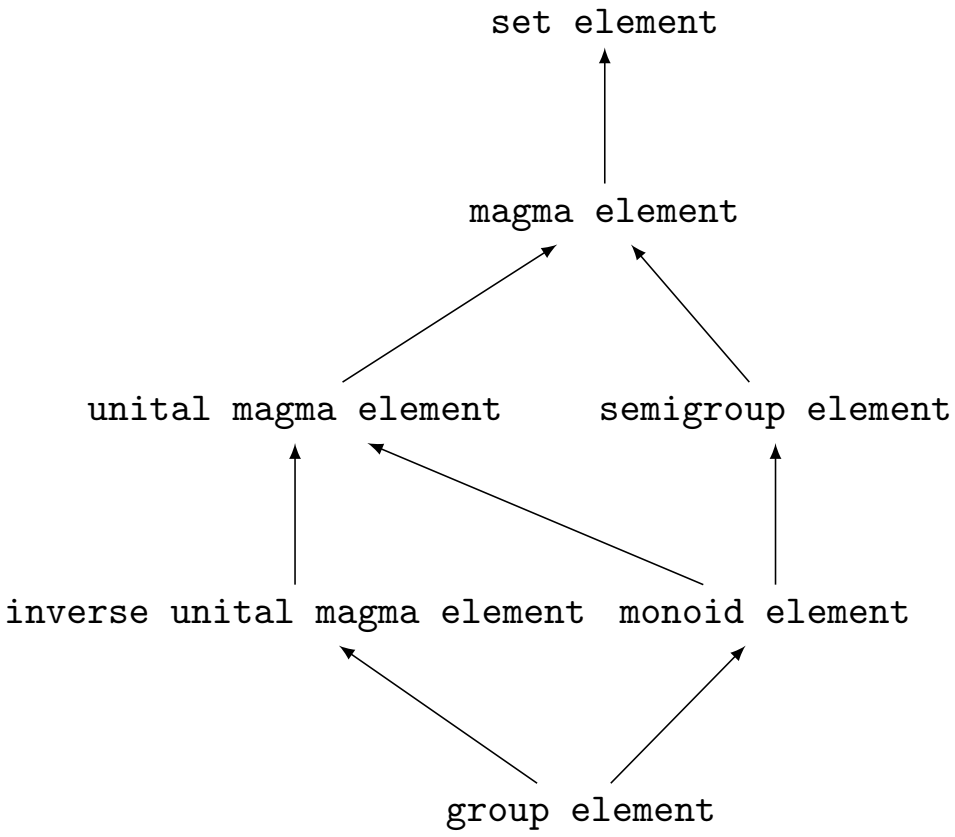
Demo



Parent, éléments, morphismes, catégories

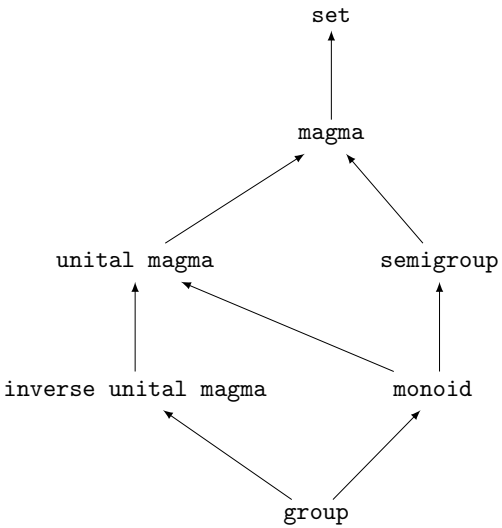
Demo

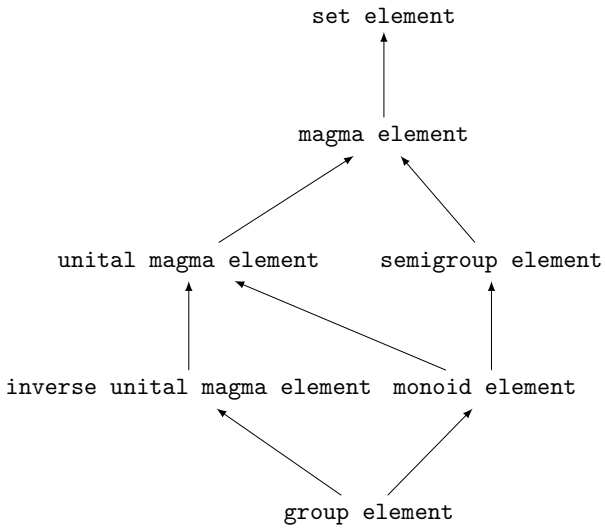


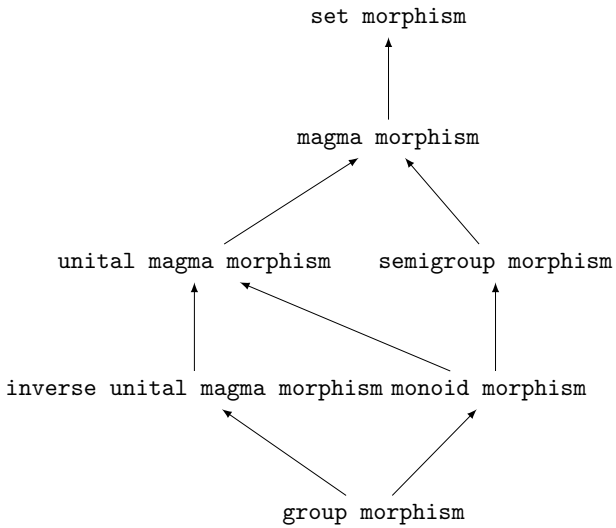


Parent, éléments, morphismes, catégories

Demo





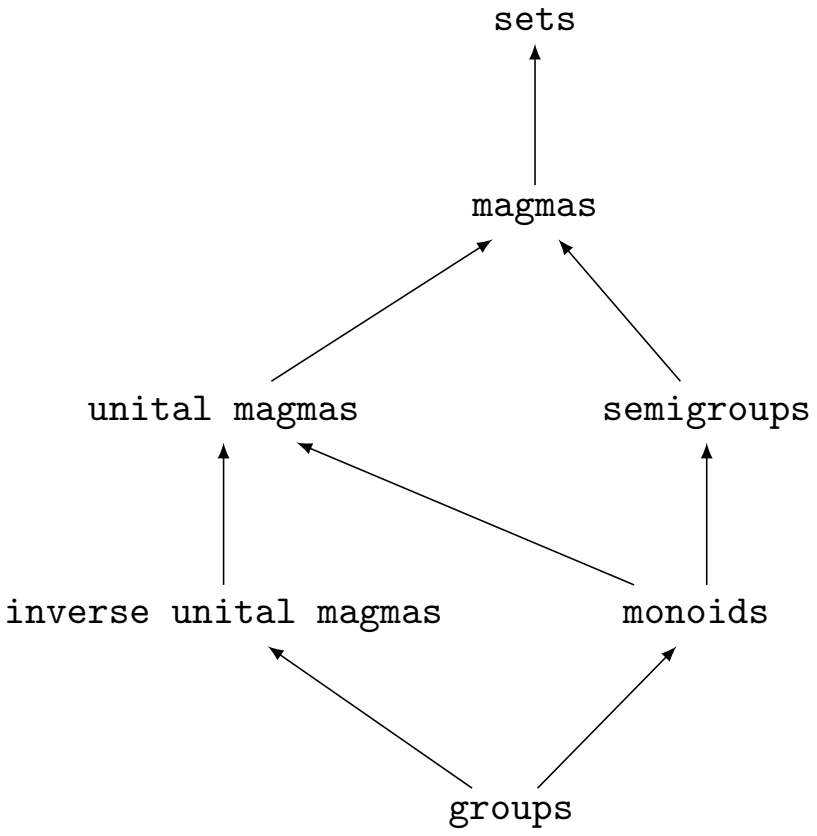


Passage à l'échelle ?

Démo

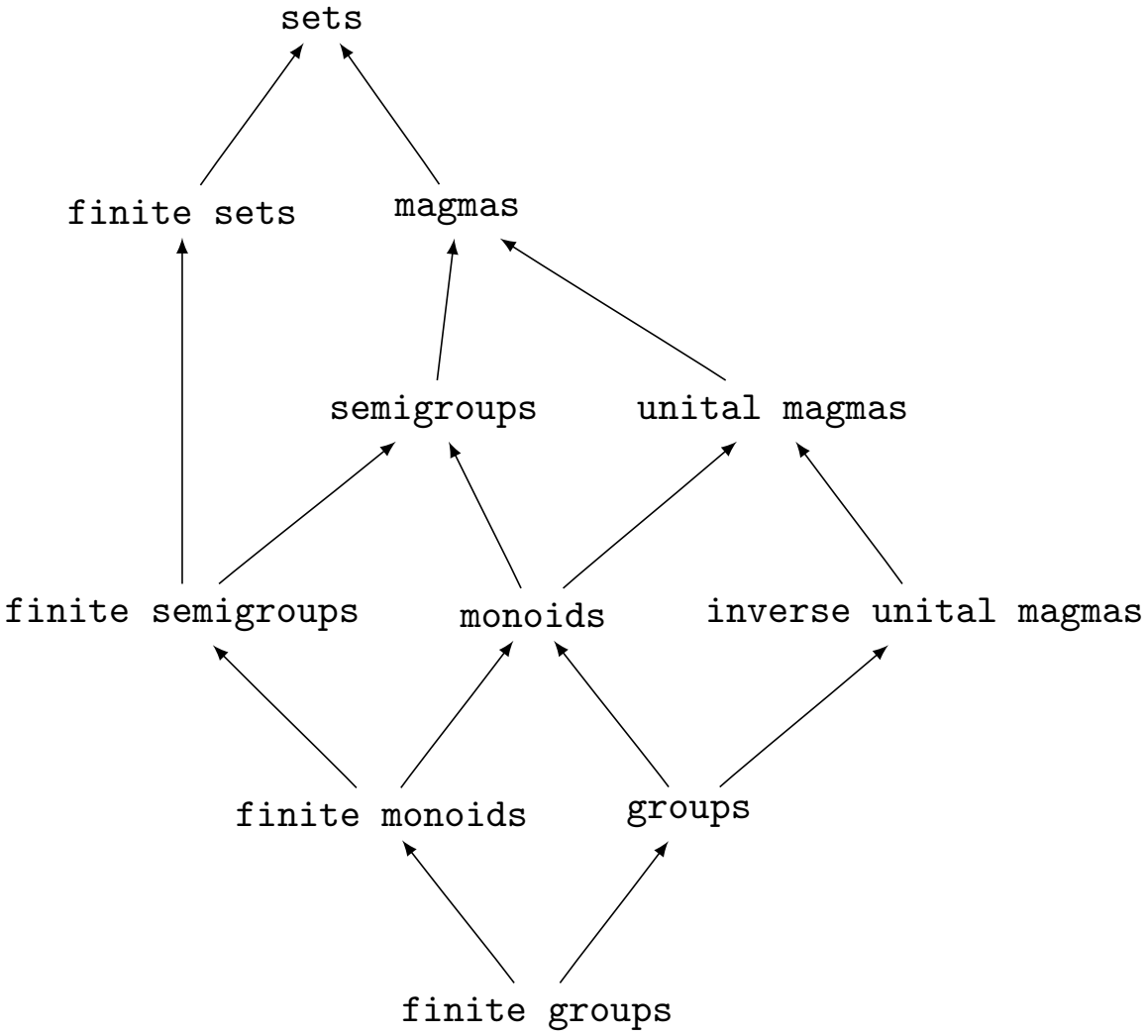
Passage à l'échelle ?

Catégories pour les groupes :



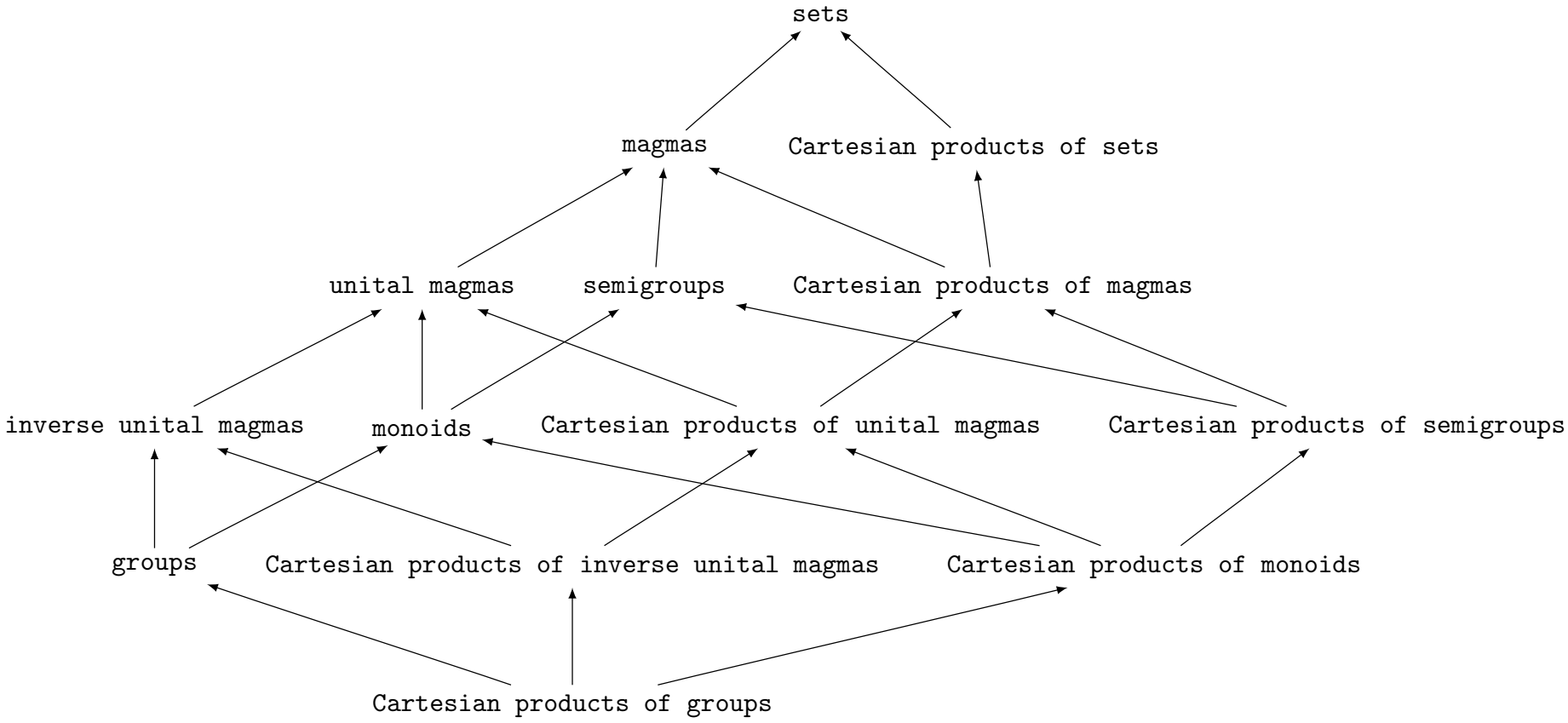
Passage à l'échelle ?

Catégories pour les groupes finis :



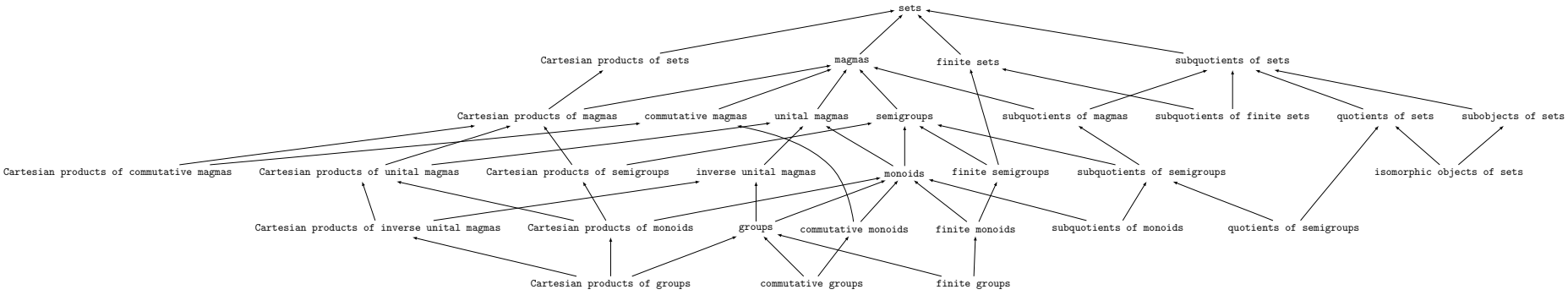
Passage à l'échelle ?

Catégories pour les produits cartésiens de groupes :



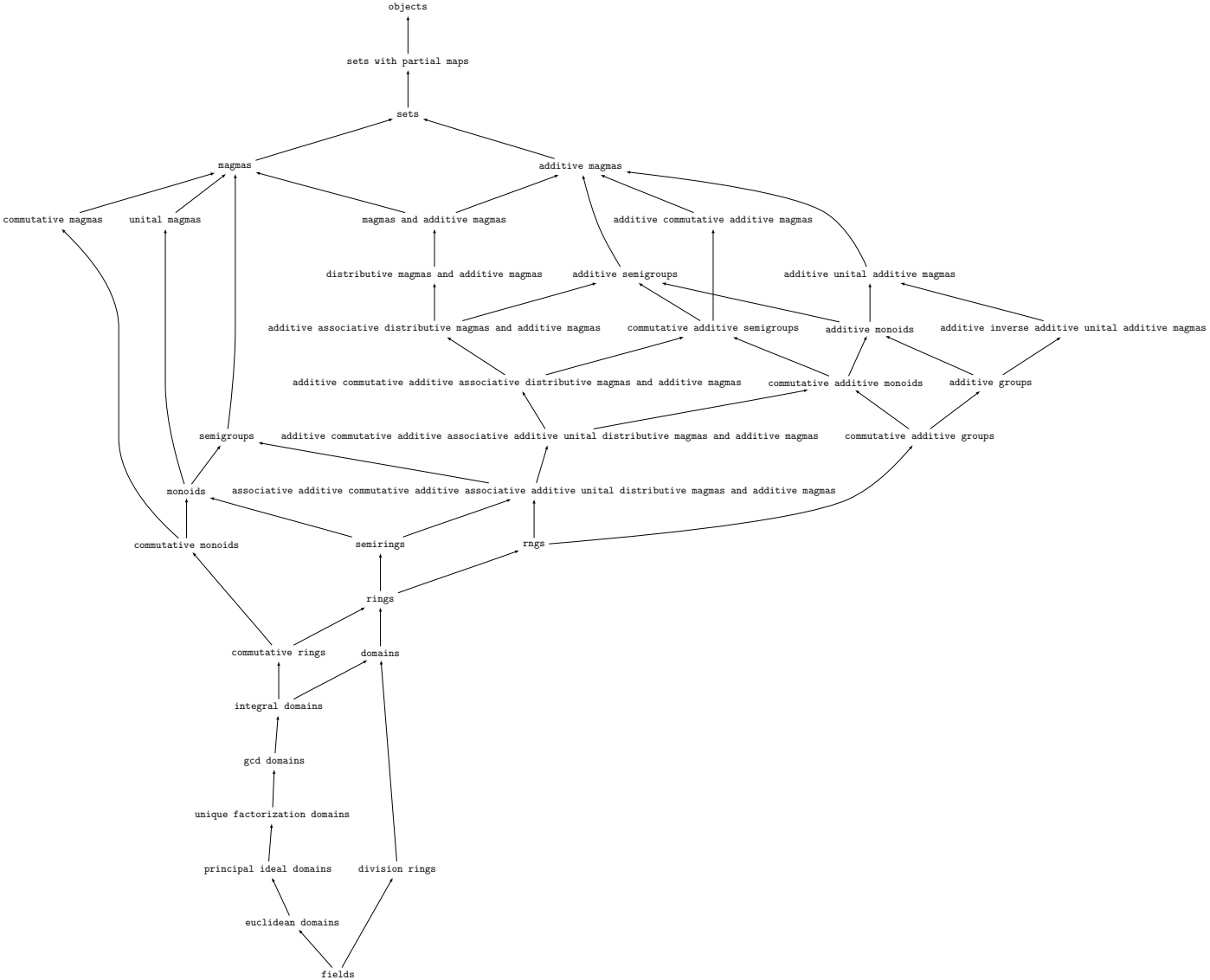
Passage à l'échelle ?

Toutes les catégories pour les groupes :



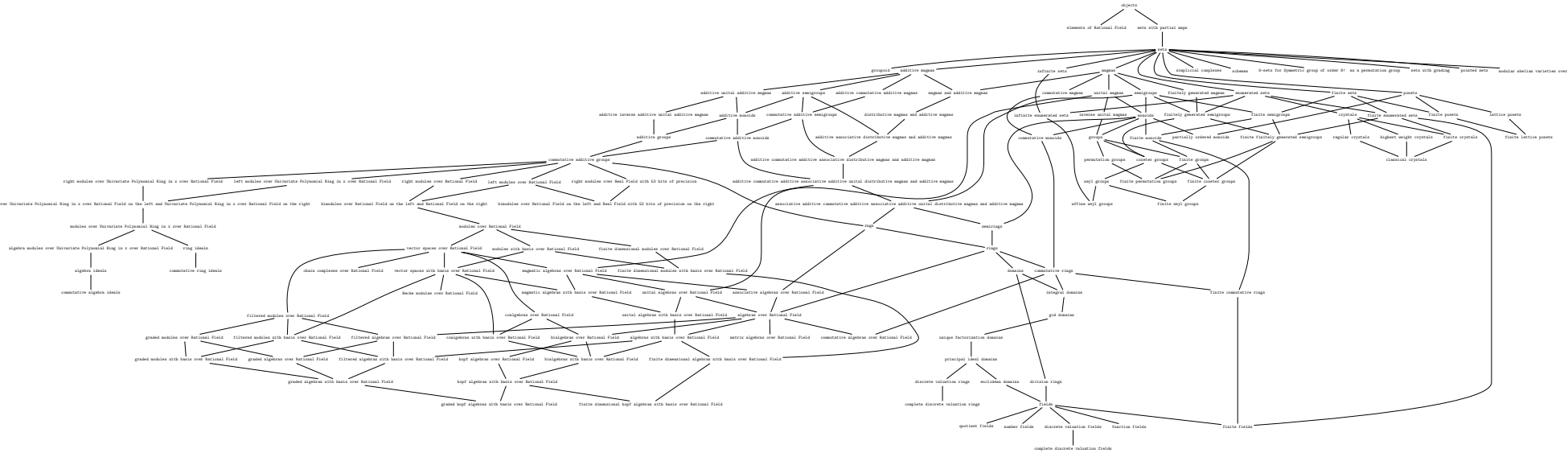
Passage à l'échelle ?

Catégories pour les corps :



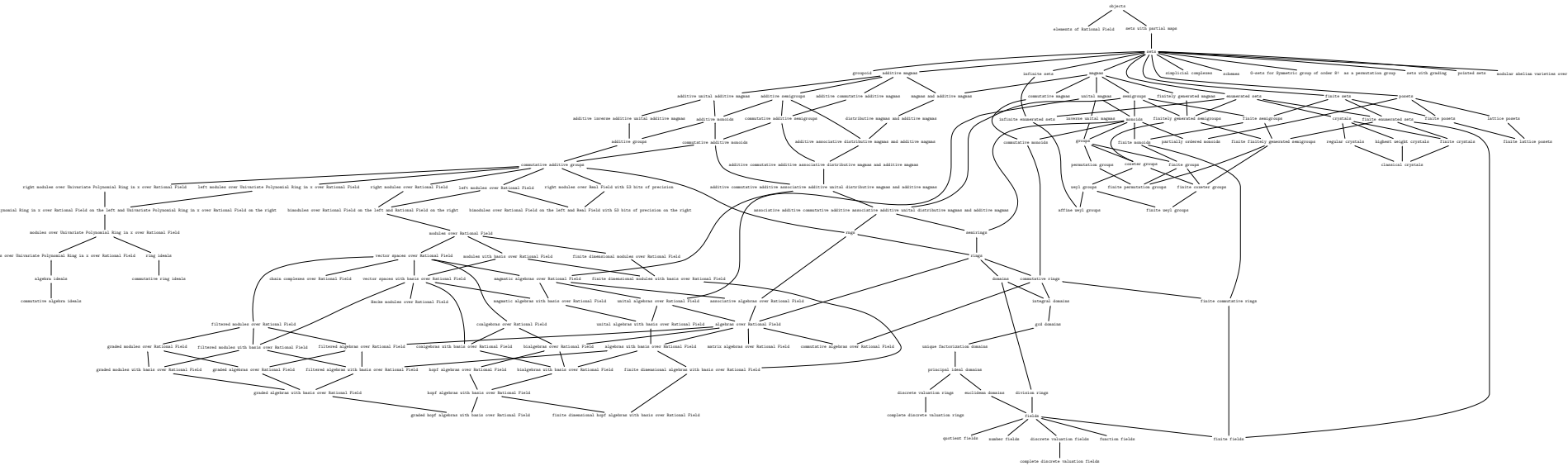
Passage à l'échelle ?

Toutes les catégories :



Passage à l'échelle ?

Toutes les catégories :



Problème

Maîtriser l'*explosion combinatoire* du nombre de classes

Résumé

- Sage modélise de très nombreux objets mathématiques
- Grande hiérarchie de catégories :
  - Sémantique
  - Mixins pour parents, éléments, morphismes
  - Documentation, Tests, Code générique
  - Constructions
- Hiérarchie robuste car elle modélise une hiérarchie existante (catégories en algèbre)
- Passage à l'échelle :
  - Construction dynamique à partir d'*information sémantique* et de *mixins* fournis par les catégories
  - Contrôle de la linéarisation

Résumé

- Sage modélise de très nombreux objets mathématiques
- Grande hiérarchie de catégories :
  - Sémantique
  - Mixins pour parents, éléments, morphismes
  - Documentation, Tests, Code générique
  - Constructions
- Hiérarchie robuste car elle modélise une hiérarchie existante (catégories en algèbre)
- Passage à l'échelle :
  - Construction dynamique à partir d'*information sémantique* et de *mixins* fournis par les catégories
  - Contrôle de la linéarisation

Le paradigme est bon ; est-ce une bonne implantation ?

Naturelle dans son contexte

- Langage dynamique (Python)
- Programmation orientée objet

En dehors de ce contexte ?

- Performances ? Compilation ?
  - Le code générique peut appeler du code compilé
  - Le code générique pourrait être compilé
  - Mais via appels de méthodes virtuelles
- Vérification statique ?

Le paradigme est bon ; est-ce une bonne implantation ?

Naturelle dans son contexte

- Langage dynamique (Python)
- Programmation orientée objet

En dehors de ce contexte ?

- Performances ? Compilation ?
  - Le code générique peut appeler du code compilé
  - Le code générique pourrait être compilé
  - Mais via appels de méthodes virtuelles
- Vérification statique ?

Pistes à explorer

Implantations alternatives du paradigme ?

- Dans un langage à typage statique ou graduel ?
- À coup de templates et de traits ?
  Par exemple en C++ / Scala
- Dans des assistants de preuve ?
  Par exemple Coq
- Gérant le multi-paramètres
  Par exemple Julia, GAP

Pistes à explorer

Modéliser plus de connaissances mathématiques ?
Formaliser des systèmes de calculs ?

- Collaboration avec des systèmes de représentation de connaissances et documents mathématiques OMDoc/MMT
- Génération automatique d'interfaces entre systèmes de calculs ?

Systèmes de documentation et de navigation ?

Bourse de thèse ou postdoc 2016-2019 au LRI !

(financée par OpenDreamKit)

Pistes à explorer

Modéliser plus de connaissances mathématiques ?
Formaliser des systèmes de calculs ?

- Collaboration avec des systèmes de représentation de connaissances et documents mathématiques OMDoc/MMT
- Génération automatique d'interfaces entre systèmes de calculs ?

Systèmes de documentation et de navigation ?

Bourse de thèse ou postdoc 2016-2019 au LRI !

(financée par OpenDreamKit)

Pistes à explorer

Modéliser plus de connaissances mathématiques ?
Formaliser des systèmes de calculs ?

- Collaboration avec des systèmes de représentation de connaissances et documents mathématiques OMDoc/MMT
- Génération automatique d'interfaces entre systèmes de calculs ?

Systèmes de documentation et de navigation ?

Bourse de thèse ou postdoc 2016-2019 au LRI !

(financée par OpenDreamKit)