Drinfeld modules in SageMath

arXiv:2305.00422

Antoine Leudière (Université de Lorraine, INRIA)

Joint work with David Ayotte, Xavier Caruso and Yossef Musleh

Wednesday July 26th, 2023

ISSAC'23

Outline of the talk

Why this project?

What is a Drinfeld module?

Focus: the crucial question of data representation

Main features

Demo

Mathematical context

Drinfeld modules:

- Introduced in the 1970s Drinfeld, 1974.
- Foundation of the class field theory for function fields.
- Function field analogues to elliptic curves
- Theory well developed and established.

Mathematical context

Drinfeld modules:

- Introduced in the 1970s Drinfeld, 1974.
- Foundation of the class field theory for function fields.
- Function field analogues to elliptic curves
- Theory well developed and established.

Mathematical context

Drinfeld modules:

- Introduced in the 1970s   Drinfeld, 1974.
- Foundation of the class field theory for function fields.
- Function field analogues to elliptic curves
- Theory well developed and established.

Mathematical context

Drinfeld modules:

- Introduced in the 1970s Drinfeld, 1974.
- Foundation of the class field theory for function fields.
- Function field analogues to elliptic curves
- Theory well developed and established.

Mathematical context

Drinfeld modules:

- Introduced in the 1970s Drinfeld, 1974.
- Foundation of the class field theory for function fields.
- Function field analogues to elliptic curves
- Theory well developed and established.

Applications and algorithmics

Applications to cryptography:

- Diffie-Hellman analogues Scanlon, 2001
- Isogeny-based cryptography Joux, Narayanan, 2019; Leudière, Spaenlehauer, 2022; Wesolowski, 2022
- Cryptanalysis of code-base cryptography Bombar, Couvreur, Debris-Alazard, 2022

Applications to computer algebra:

- Efficient factorization in $\mathbb{F}_q[X]$ Doliskani, Narayanan, Schost, 2021, .

Algorithmics:

- Isogenies Caranay, 2018 (thesis).
- Characteristic polynomials of endomorphisms and norms of isogenies Musleh, Schost, **ISSAC** 2019; Musleh, Schost, **ISSAC 2023**; Caruso, Leudière, 2023 (preprint).
- Isogenies and modular polynomials: Caranay, Greenberg, Scheidler, 2020.
- Class field theory: Leudière, Spaenlehauer, 2021 (preprint).

Applications and algorithmics

Applications to cryptography:

- Diffie-Hellman analogues Scanlon, 2001
- Isogeny-based cryptography Joux, Narayanan, 2019; Leudière, Spaenlehauer, 2022; Wesolowski, 2022
- Cryptanalysis of code-base cryptography Bombar, Couvreur, Debris-Alazard, 2022

Applications to computer algebra:

- Efficient factorization in $\mathbb{F}_q[X]$ Doliskani, Narayanan, Schost, 2021, .

Algorithmics:

- Isogenies Caranay, 2018 (thesis).
- Characteristic polynomials of endomorphisms and norms of isogenies Musleh, Schost, **ISSAC** 2019; Musleh, Schost, **ISSAC 2023**; Caruso, Leudière, 2023 (preprint).
- Isogenies and modular polynomials: Caranay, Greenberg, Scheidler, 2020.
- Class field theory: Leudière, Spaenlehauer, 2021 (preprint).

Applications and algorithmics

Applications to cryptography:

- Diffie-Hellman analogues Scanlon, 2001
- Isogeny-based cryptography Joux, Narayanan, 2019; Leudière, Spaenlehauer, 2022; Wesolowski, 2022
- Cryptanalysis of code-base cryptography Bombar, Couvreur, Debris-Alazard, 2022

Applications to computer algebra:

- Efficient factorization in $\mathbb{F}_q[X]$ Doliskani, Narayanan, Schost, 2021, .

Algorithmics:

- Isogenies Caranay, 2018 (thesis).
- Characteristic polynomials of endomorphisms and norms of isogenies Musleh, Schost, **ISSAC** 2019; Musleh, Schost, **ISSAC 2023**; Caruso, Leudière, 2023 (preprint).
- Isogenies and modular polynomials: Caranay, Greenberg, Scheidler, 2020.
- Class field theory: Leudière, Spaenlehauer, 2021 (preprint).

Applications and algorithmics

Applications to cryptography:

- Diffie-Hellman analogues   Scanlon, 2001
- Isogeny-based cryptography   Joux, Narayanan, 2019; Leudière, Spaenlehauer, 2022; Wesolowski, 2022
- Cryptanalysis of code-base cryptography   Bombar, Couvreur, Debris-Alazard, 2022

Applications to computer algebra:

- Efficient factorization in $\mathbb{F}_q[X]$ Doliskani, Narayanan, Schost, 2021, .

Algorithmics:

- Isogenies   Caranay, 2018 (thesis).
- Characteristic polynomials of endomorphisms and norms of isogenies   Musleh, Schost, **ISSAC** 2019; Musleh, Schost, **ISSAC 2023**; Caruso, Leudière, 2023 (preprint).
- Isogenies and modular polynomials: Caranay, Greenberg, Scheidler, 2020.
- Class field theory: Leudière, Spaenlehauer, 2021 (preprint).

Applications and algorithmics

Applications to cryptography:

- Diffie-Hellman analogues Scanlon, 2001
- Isogeny-based cryptography Joux, Narayanan, 2019; Leudière, Spaenlehauer, 2022; Wesolowski, 2022
- Cryptanalysis of code-base cryptography Bombar, Couvreur, Debris-Alazard, 2022

Applications to computer algebra:

- Efficient factorization in $\mathbb{F}_q[X]$ Doliskani, Narayanan, Schost, 2021, .

Algorithmics:

- Isogenies Caranay, 2018 (thesis).
- Characteristic polynomials of endomorphisms and norms of isogenies Musleh, Schost, **ISSAC** 2019; Musleh, Schost, **ISSAC 2023**; Caruso, Leudière, 2023 (preprint).
- Isogenies and modular polynomials: Caranay, Greenberg, Scheidler, 2020.
- Class field theory: Leudière, Spaenlehauer, 2021 (preprint).

Applications and algorithmics

Applications to cryptography:

- Diffie-Hellman analogues Scanlon, 2001
- Isogeny-based cryptography Joux, Narayanan, 2019; Leudière, Spaenlehauer, 2022; Wesolowski, 2022
- Cryptanalysis of code-base cryptography Bombar, Couvreur, Debris-Alazard, 2022

Applications to computer algebra:

- Efficient factorization in $\mathbb{F}_q[X]$ Doliskani, Narayanan, Schost, 2021, .

Algorithmics:

- Isogenies Caranay, 2018 (thesis).
- Characteristic polynomials of endomorphisms and norms of isogenies Musleh, Schost, **ISSAC** 2019; Musleh, Schost, **ISSAC 2023**; Caruso, Leudière, 2023 (preprint).
- Isogenies and modular polynomials: Caranay, Greenberg, Scheidler, 2020.
- Class field theory: Leudière, Spaenlehauer, 2021 (preprint).

Applications and algorithmics

Applications to cryptography:

- Diffie-Hellman analogues Scanlon, 2001
- Isogeny-based cryptography Joux, Narayanan, 2019; Leudière, Spaenlehauer, 2022; Wesolowski, 2022
- Cryptanalysis of code-base cryptography Bombar, Couvreur, Debris-Alazard, 2022

Applications to computer algebra:

- Efficient factorization in $\mathbb{F}_q[X]$ Doliskani, Narayanan, Schost, 2021, .

Algorithmics:

- Isogenies Caranay, 2018 (thesis).
- Characteristic polynomials of endomorphisms and norms of isogenies Musleh, Schost, **ISSAC** 2019; Musleh, Schost, **ISSAC 2023**; Caruso, Leudière, 2023 (preprint).
- Isogenies and modular polynomials: Caranay, Greenberg, Scheidler, 2020.
- Class field theory: Leudière, Spaenlehauer, 2021 (preprint).

Applications and algorithmics

Applications to cryptography:

- Diffie-Hellman analogues   Scanlon, 2001
- Isogeny-based cryptography   Joux, Narayanan, 2019; Leudière, Spaenlehauer, 2022; Wesolowski, 2022
- Cryptanalysis of code-base cryptography   Bombar, Couvreur, Debris-Alazard, 2022

Applications to computer algebra:

- Efficient factorization in $\mathbb{F}_q[X]$ Doliskani, Narayanan, Schost, 2021, .

Algorithmics:

- Isogenies   Caranay, 2018 (thesis).
- Characteristic polynomials of endomorphisms and norms of isogenies   Musleh, Schost, **ISSAC** 2019; Musleh, Schost, **ISSAC 2023**; Caruso, Leudière, 2023 (preprint).
- Isogenies and modular polynomials: Caranay, Greenberg, Scheidler, 2020.
- Class field theory: Leudière, Spaenlehauer, 2021 (preprint).

Applications and algorithmics

Applications to cryptography:

- Diffie-Hellman analogues Scanlon, 2001
- Isogeny-based cryptography Joux, Narayanan, 2019; Leudière, Spaenlehauer, 2022; Wesolowski, 2022
- Cryptanalysis of code-base cryptography Bombar, Couvreur, Debris-Alazard, 2022

Applications to computer algebra:

- Efficient factorization in $\mathbb{F}_q[X]$ Doliskani, Narayanan, Schost, 2021, .

Algorithmics:

- Isogenies Caranay, 2018 (thesis).
- Characteristic polynomials of endomorphisms and norms of isogenies Musleh, Schost, **ISSAC** 2019; Musleh, Schost, **ISSAC 2023**; Caruso, Leudière, 2023 (preprint).
- Isogenies and modular polynomials: Caranay, Greenberg, Scheidler, 2020.
- Class field theory: Leudière, Spaenlehauer, 2021 (preprint).

Applications and algorithmics

Applications to cryptography:

- Diffie-Hellman analogues Scanlon, 2001
- Isogeny-based cryptography Joux, Narayanan, 2019; Leudière, Spaenlehauer, 2022; Wesolowski, 2022
- Cryptanalysis of code-base cryptography Bombar, Couvreur, Debris-Alazard, 2022

Applications to computer algebra:

- Efficient factorization in $\mathbb{F}_q[X]$ Doliskani, Narayanan, Schost, 2021, .

Algorithmics:

- Isogenies Caranay, 2018 (thesis).
- Characteristic polynomials of endomorphisms and norms of isogenies Musleh, Schost, **ISSAC** 2019; Musleh, Schost, **ISSAC 2023**; Caruso, Leudière, 2023 (preprint).
- Isogenies and modular polynomials: Caranay, Greenberg, Scheidler, 2020.
- Class field theory: Leudière, Spaenlehauer, 2021 (preprint).

Why this implementation?

We want to help mathematicians using Drinfeld modules.

- Drinfeld modules are very abstract project with no graphical representation.
- Develop intuition.
- Create conjectures.
- Test conjectures and create databases Hayes, 1994.

SageMath benefits:

- SageMath reaches numerous and various mathematicians.
- Benefit from Free and Open Source Software.
- Elementary building blocks were already in SageMath.

Why this implementation?

We want to help mathematicians using Drinfeld modules.

- Drinfeld modules are very abstract project with no graphical representation.
- Develop intuition.
- Create conjectures.
- Test conjectures and create databases Hayes, 1994.

SageMath benefits:

- SageMath reaches numerous and various mathematicians.
- Benefit from Free and Open Source Software.
- Elementary building blocks were already in SageMath.

Why this implementation?

We want to help mathematicians using Drinfeld modules.

- Drinfeld modules are very abstract project with no graphical representation.
- Develop intuition.
- Create conjectures.
- Test conjectures and create databases Hayes, 1994.

SageMath benefits:

- SageMath reaches numerous and various mathematicians.
- Benefit from Free and Open Source Software.
- Elementary building blocks were already in SageMath.

Why this implementation?

We want to help mathematicians using Drinfeld modules.

- Drinfeld modules are very abstract project with no graphical representation.
- Develop intuition.
- Create conjectures.
- Test conjectures and create databases Hayes, 1994.

SageMath benefits:

- SageMath reaches numerous and various mathematicians.
- Benefit from Free and Open Source Software.
- Elementary building blocks were already in SageMath.

Why this implementation?

We want to help mathematicians using Drinfeld modules.

- Drinfeld modules are very abstract project with no graphical representation.
- Develop intuition.
- Create conjectures.
- Test conjectures and create databases Hayes, 1994.

SageMath benefits:

- SageMath reaches numerous and various mathematicians.
- Benefit from Free and Open Source Software.
- Elementary building blocks were already in SageMath.

Why this implementation?

We want to help mathematicians using Drinfeld modules.

- Drinfeld modules are very abstract project with no graphical representation.
- Develop intuition.
- Create conjectures.
- Test conjectures and create databases Hayes, 1994.

SageMath benefits:

- SageMath reaches numerous and various mathematicians.
- Benefit from Free and Open Source Software.
- Elementary building blocks were already in SageMath.

Why this implementation?

We want to help mathematicians using Drinfeld modules.

- Drinfeld modules are very abstract project with no graphical representation.
- Develop intuition.
- Create conjectures.
- Test conjectures and create databases Hayes, 1994.

SageMath benefits:

- SageMath reaches numerous and various mathematicians.
- Benefit from Free and Open Source Software.
- Elementary building blocks were already in SageMath.

Why this implementation?

We want to help mathematicians using Drinfeld modules.

- Drinfeld modules are very abstract project with no graphical representation.
- Develop intuition.
- Create conjectures.
- Test conjectures and create databases Hayes, 1994.

SageMath benefits:

- SageMath reaches numerous and various mathematicians.
- Benefit from Free and Open Source Software.
- Elementary building blocks were already in SageMath.

Why this implementation?

We want to help mathematicians using Drinfeld modules.

- Drinfeld modules are very abstract project with no graphical representation.
- Develop intuition.
- Create conjectures.
- Test conjectures and create databases Hayes, 1994.

SageMath benefits:

- SageMath reaches numerous and various mathematicians.
- Benefit from Free and Open Source Software.
- Elementary building blocks were already in SageMath.

Outline of the talk

Why this project?

What is a Drinfeld module?

Focus: the crucial question of data representation

Main features

Demo

Definition: algebraic structure on geometric objects

$$\mathbb{F}_q[T] \xrightarrow{\gamma} K$$

Function field

Finite field

A Drinfeld module endows $\overline{K}$ with a structure of $\mathbb{F}_q[T]$-module.

Definition

A Drinfeld $\mathbb{F}_q[T]$-module over K is an $\mathbb{F}_q$-algebra morphism (satisfying extra conditions)

$$\phi : \mathbb{F}_q[T] \to \{f \in \mathrm{End}_{\mathbb{F}_q}(\overline{K}) \text{ defined over } K\} = \mathrm{Span}_K((\tau^i : x \mapsto x^{q^i})_{i \in \mathbb{Z}_{\geqslant 0}}) = K\{\tau\}.$$

Definition: algebraic structure on geometric objects

$$\mathbb{F}_q[T] \xrightarrow{\gamma} K$$

Function field

Finite field

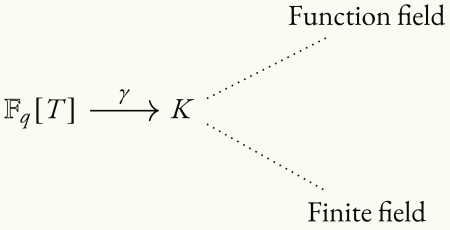
A Drinfeld module endows $\overline{K}$ with a structure of $\mathbb{F}_q[T]$-module.

Definition

A Drinfeld $\mathbb{F}_q[T]$-module over K is an $\mathbb{F}_q$-algebra morphism (satisfying extra conditions)

$$\phi : \mathbb{F}_q[T] \to \{f \in \mathrm{End}_{\mathbb{F}_q}(\overline{K}) \text{ defined over } K\} = \mathrm{Span}_K((\tau^i : x \mapsto x^{q^i})_{i \in \mathbb{Z}_{\geqslant 0}}) = K\{\tau\}.$$

Definition: algebraic structure on geometric objects



A Drinfeld module endows $\overline{K}$ with a structure of $\mathbb{F}_q[T]$-module.

Definition

A Drinfeld $\mathbb{F}_q[T]$-module over K is an $\mathbb{F}_q$-algebra morphism (satisfying extra conditions)

$$\phi : \mathbb{F}_q[T] \to \{f \in \mathrm{End}_{\mathbb{F}_q}(\overline{K}) \text{ defined over } K\} = \mathrm{Span}_K((\tau^i : x \mapsto x^{q^i})_{i \in \mathbb{Z}_{\geqslant 0}}) = K\{\tau\}.$$

Outline of the talk

Why this project?

What is a Drinfeld module?

Focus: the crucial question of data representation

Main features

Demo

Representation of Drinfeld modules

A Drinfeld module $\phi : \mathbb{F}_q[T] \to K\{\tau\}$ can be represented by:

- A morphism.
- A skew polynomial $\phi(T) = g_0 + g_1\tau + \cdots + g_r\tau^r \in K\{\tau\}$.

A Drinfeld module is *not* a set!

Representation of Drinfeld modules

A Drinfeld module $\phi : \mathbb{F}_q[T] \to K\{\tau\}$ can be represented by:

- A morphism.
- A skew polynomial $\phi(T) = g_0 + g_1\tau + \cdots + g_r\tau^r \in K\{\tau\}$.

A Drinfeld module is *not* a set!

Representation of Drinfeld modules

A Drinfeld module $\phi : \mathbb{F}_q[T] \to K\{\tau\}$ can be represented by:

- A morphism.
- A skew polynomial $\phi(T) = g_0 + g_1\tau + \cdots + g_r\tau^r \in K\{\tau\}$.

A Drinfeld module is *not* a set!

Representation of Drinfeld modules

A Drinfeld module $\phi : \mathbb{F}_q[T] \to K\{\tau\}$ can be represented by:

- A morphism.
- A skew polynomial $\phi(T) = g_0 + g_1\tau + \cdots + g_r\tau^r \in K\{\tau\}$.

A Drinfeld module is *not* a set!

The Parent/Element framework

Parent/Element framework

Every object is either:

- a set (`Parent`);
- an element in the set (`Element`);
- a category whose objects are `Parents`.

Drinfeld modules do not really fit.

- Drinfeld modules should be objects in a category, so `Parents`.
- Drinfeld modules are not sets, so should not be `Parents`.

The Parent/Element framework

Parent/Element framework

Every object is either:

- a set (`Parent`);
- an element in the set (`Element`);
- a category whose objects are `Parents`.

Drinfeld modules do not really fit.

- Drinfeld modules should be objects in a category, so `Parents`.
- Drinfeld modules are not sets, so should not be `Parents`.

The Parent/Element framework

Parent/Element framework

Every object is either:

- a set (`Parent`);
- an element in the set (`Element`);
- a category whose objects are `Parent`s.

Drinfeld modules do not really fit.

- Drinfeld modules should be objects in a category, so `Parent`s.
- Drinfeld modules are not sets, so should not be `Parent`s.

The Parent/Element framework

Parent/Element framework

Every object is either:

- a set (`Parent`);
- an element in the set (`Element`);
- a category whose objects are `Parents`.

Drinfeld modules do not really fit.

- Drinfeld modules should be objects in a category, so `Parents`.
- Drinfeld modules are not sets, so should not be `Parents`.

The Parent/Element framework

Parent/Element framework

Every object is either:

- a set (`Parent`);
- an element in the set (`Element`);
- a category whose objects are `Parents`.

Drinfeld modules do not really fit.

- Drinfeld modules should be objects in a category, so `Parents`.
- Drinfeld modules are not sets, so should not be `Parents`.

The Parent/Element framework

Parent/Element framework

Every object is either:

- a set (`Parent`);
- an element in the set (`Element`);
- a category whose objects are `Parents`.

Drinfeld modules do not really fit.

- Drinfeld modules should be objects in a category, so `Parents`.
- Drinfeld modules are not sets, so should not be `Parents`.

The Parent/Element framework

Parent/Element framework

Every object is either:

- a set (`Parent`);
- an element in the set (`Element`);
- a category whose objects are `Parent`s.

Drinfeld modules do not really fit.

- Drinfeld modules should be objects in a category, so `Parent`s.
- Drinfeld modules are not sets, so should not be `Parent`s.

Possible solutions

1. Making Drinfeld modules `Parents` without `Elements`.
    - Strong mathematical soundness.
    - Follow `EllipticCurve`.
    - Drawback 1: `Parents` should have `Elements`.
    - Drawback 2: the category of a `Parent` must be a subcategory of **Sets**.
2. Making Drinfeld modules a `CategoryObject`.
    - Drawback: barely used in the codebase.
3. Making Drinfeld modules `Elements` and their category a `Parent`.
    - Drawback 1: the category of Drinfeld modules should be a proper `Category`.
    - Drawback 2: technical difficulties for the implementation of morphisms.

After a passionate debate, we made Drinfeld modules `Parents` without `Elements`.

Possible solutions

1. Making Drinfeld modules `Parents` without `Elements`.
   - Strong mathematical soundness.
   - Follow `EllipticCurve`.
   - Drawback 1: `Parents` should have `Elements`.
   - Drawback 2: the category of a `Parent` must be a subcategory of **Sets**.
2. Making Drinfeld modules a `CategoryObject`.
   - Drawback: barely used in the codebase.
3. Making Drinfeld modules `Elements` and their category a `Parent`.
   - Drawback 1: the category of Drinfeld modules should be a proper `Category`.
   - Drawback 2: technical difficulties for the implementation of morphisms.

After a passionate debate, we made Drinfeld modules `Parents` without `Elements`.

Possible solutions

1. Making Drinfeld modules `Parents` without `Elements`.
    - Strong mathematical soundness.
    - Follow `EllipticCurve`.
    - Drawback 1: `Parents` should have `Elements`.
    - Drawback 2: the category of a `Parent` must be a subcategory of **Sets**.
2. Making Drinfeld modules a `CategoryObject`.
    - Drawback: barely used in the codebase.
3. Making Drinfeld modules `Elements` and their category a `Parent`.
    - Drawback 1: the category of Drinfeld modules should be a proper `Category`.
    - Drawback 2: technical difficulties for the implementation of morphisms.

After a passionate debate, we made Drinfeld modules `Parents` without `Elements`.

Possible solutions

1. Making Drinfeld modules `Parents` without `Elements`.
   - Strong mathematical soundness.
   - Follow `EllipticCurve`.
   - Drawback 1: `Parents` should have `Elements`.
   - Drawback 2: the category of a `Parent` must be a subcategory of **Sets**.
2. Making Drinfeld modules a `CategoryObject`.
   - Drawback: barely used in the codebase.
3. Making Drinfeld modules `Elements` and their category a `Parent`.
   - Drawback 1: the category of Drinfeld modules should be a proper `Category`.
   - Drawback 2: technical difficulties for the implementation of morphisms.

After a passionate debate, we made Drinfeld modules `Parents` without `Elements`.

Possible solutions

1. Making Drinfeld modules `Parents` without `Elements`.
   - Strong mathematical soundness.
   - Follow `EllipticCurve`.
   - Drawback 1: `Parents` should have `Elements`.
   - Drawback 2: the category of a `Parent` must be a subcategory of **Sets**.
2. Making Drinfeld modules a `CategoryObject`.
   - Drawback: barely used in the codebase.
3. Making Drinfeld modules `Elements` and their category a `Parent`.
   - Drawback 1: the category of Drinfeld modules should be a proper `Category`.
   - Drawback 2: technical difficulties for the implementation of morphisms.

After a passionate debate, we made Drinfeld modules `Parents` without `Elements`.

Possible solutions

1. Making Drinfeld modules `Parents` without `Elements`.
    - Strong mathematical soundness.
    - Follow `EllipticCurve`.
    - Drawback 1: `Parents` should have `Elements`.
    - Drawback 2: the category of a `Parent` must be a subcategory of **Sets**.
2. Making Drinfeld modules a `CategoryObject`.
    - Drawback: barely used in the codebase.
3. Making Drinfeld modules `Elements` and their category a `Parent`.
    - Drawback 1: the category of Drinfeld modules should be a proper `Category`.
    - Drawback 2: technical difficulties for the implementation of morphisms.

After a passionate debate, we made Drinfeld modules `Parents` without `Elements`.

Possible solutions

1. Making Drinfeld modules `Parents` without `Elements`.
    - Strong mathematical soundness.
    - Follow `EllipticCurve`.
    - Drawback 1: `Parents` should have `Elements`.
    - Drawback 2: the category of a `Parent` must be a subcategory of **Sets**.
2. Making Drinfeld modules a `CategoryObject`.
    - Drawback: barely used in the codebase.
3. Making Drinfeld modules `Elements` and their category a `Parent`.
    - Drawback 1: the category of Drinfeld modules should be a proper `Category`.
    - Drawback 2: technical difficulties for the implementation of morphisms.

After a passionate debate, we made Drinfeld modules `Parents` without `Elements`.

Possible solutions

1. Making Drinfeld modules `Parents` without `Elements`.
   - Strong mathematical soundness.
   - Follow `EllipticCurve`.
   - Drawback 1: `Parents` should have `Elements`.
   - Drawback 2: the category of a `Parent` must be a subcategory of **Sets**.
2. Making Drinfeld modules a `CategoryObject`.
   - Drawback: barely used in the codebase.
3. Making Drinfeld modules `Elements` and their category a `Parent`.
   - Drawback 1: the category of Drinfeld modules should be a proper `Category`.
   - Drawback 2: technical difficulties for the implementation of morphisms.

After a passionate debate, we made Drinfeld modules `Parents` without `Elements`.

Possible solutions

1. Making Drinfeld modules `Parents` without `Elements`.
   - Strong mathematical soundness.
   - Follow `EllipticCurve`.
   - Drawback 1: `Parents` should have `Elements`.
   - Drawback 2: the category of a `Parent` must be a subcategory of **Sets**.
2. Making Drinfeld modules a `CategoryObject`.
   - Drawback: barely used in the codebase.
3. Making Drinfeld modules `Elements` and their category a `Parent`.
   - Drawback 1: the category of Drinfeld modules should be a proper `Category`.
   - Drawback 2: technical difficulties for the implementation of morphisms.

After a passionate debate, we made Drinfeld modules `Parents` without `Elements`.

Possible solutions

1. Making Drinfeld modules `Parents` without `Elements`.
    - Strong mathematical soundness.
    - Follow `EllipticCurve`.
    - Drawback 1: `Parents` should have `Elements`.
    - Drawback 2: the category of a `Parent` must be a subcategory of **Sets**.
2. Making Drinfeld modules a `CategoryObject`.
    - Drawback: barely used in the codebase.
3. Making Drinfeld modules `Elements` and their category a `Parent`.
    - Drawback 1: the category of Drinfeld modules should be a proper `Category`.
    - Drawback 2: technical difficulties for the implementation of morphisms.

After a passionate debate, we made Drinfeld modules `Parents` without `Elements`.

Possible solutions

1. Making Drinfeld modules `Parents` without `Elements`.
   - Strong mathematical soundness.
   - Follow `EllipticCurve`.
   - Drawback 1: `Parents` should have `Elements`.
   - Drawback 2: the category of a `Parent` must be a subcategory of **Sets**.
2. Making Drinfeld modules a `CategoryObject`.
   - Drawback: barely used in the codebase.
3. Making Drinfeld modules `Elements` and their category a `Parent`.
   - Drawback 1: the category of Drinfeld modules should be a proper `Category`.
   - Drawback 2: technical difficulties for the implementation of morphisms.

After a passionate debate, we made Drinfeld modules `Parents` without `Elements`.

Outline of the talk

Why this project?

What is a Drinfeld module?

Focus: the crucial question of data representation

Main features

Demo

Main features

Features:

- General constructions (Drinfeld modules, morphisms, category).
- Basic computations (evaluation, rank, height, j-invariant, action on $\overline{K}$).
- Morphism computations (action on *homsets*, Velu, generalized j-invariants, characteristic polynomials of endomorphisms and norms of isogenies).
- Analytic construction of Drinfeld modules (logarithm and exponential).

User-oriented design:

- Simple, high-level, elegant interface.
- Exhaustive, useful documentation.
- Thorough testing.
- The development is still active, with contributions welcome.
- We had great feedback from the community.

First features were released in SageMath 10.0. The rest is being reviewed.

Main features

Features:

- General constructions (Drinfeld modules, morphisms, category).
- Basic computations (evaluation, rank, height, j-invariant, action on $\overline{K}$).
- Morphism computations (action on *homsets*, Velu, generalized j-invariants, characteristic polynomials of endomorphisms and norms of isogenies).
- Analytic construction of Drinfeld modules (logarithm and exponential).

User-oriented design:

- Simple, high-level, elegant interface.
- Exhaustive, useful documentation.
- Thorough testing.
- The development is still active, with contributions welcome.
- We had great feedback from the community.

First features were released in SageMath 10.0. The rest is being reviewed.

Main features

Features:

- General constructions (Drinfeld modules, morphisms, category).
- Basic computations (evaluation, rank, height, j-invariant, action on $\overline{K}$).
- Morphism computations (action on *homsets*, Velu, generalized j-invariants, characteristic polynomials of endomorphisms and norms of isogenies).
- Analytic construction of Drinfeld modules (logarithm and exponential).

User-oriented design:

- Simple, high-level, elegant interface.
- Exhaustive, useful documentation.
- Thorough testing.
- The development is still active, with contributions welcome.
- We had great feedback from the community.

First features were released in SageMath 10.0. The rest is being reviewed.

Main features

Features:

- General constructions (Drinfeld modules, morphisms, category).
- Basic computations (evaluation, rank, height, j-invariant, action on $\overline{K}$).
- Morphism computations (action on *homsets*, Velu, generalized j-invariants, characteristic polynomials of endomorphisms and norms of isogenies).
- Analytic construction of Drinfeld modules (logarithm and exponential).

User-oriented design:

- Simple, high-level, elegant interface.
- Exhaustive, useful documentation.
- Thorough testing.
- The development is still active, with contributions welcome.
- We had great feedback from the community.

First features were released in SageMath 10.0. The rest is being reviewed.

Main features

Features:

- General constructions (Drinfeld modules, morphisms, category).
- Basic computations (evaluation, rank, height, j-invariant, action on $\overline{K}$).
- Morphism computations (action on *homsets*, Velu, generalized j-invariants, characteristic polynomials of endomorphisms and norms of isogenies).
- Analytic construction of Drinfeld modules (logarithm and exponential).

User-oriented design:

- Simple, high-level, elegant interface.
- Exhaustive, useful documentation.
- Thorough testing.
- The development is still active, with contributions welcome.
- We had great feedback from the community.

First features were released in SageMath 10.0. The rest is being reviewed.

Main features

Features:

- General constructions (Drinfeld modules, morphisms, category).
- Basic computations (evaluation, rank, height, j-invariant, action on $\overline{K}$).
- Morphism computations (action on *homsets*, Velu, generalized j-invariants, characteristic polynomials of endomorphisms and norms of isogenies).
- Analytic construction of Drinfeld modules (logarithm and exponential).

User-oriented design:

- Simple, high-level, elegant interface.
- Exhaustive, useful documentation.
- Thorough testing.
- The development is still active, with contributions welcome.
- We had great feedback from the community.

First features were released in SageMath 10.0. The rest is being reviewed.

Main features

Features:

- General constructions (Drinfeld modules, morphisms, category).
- Basic computations (evaluation, rank, height, j-invariant, action on $\overline{K}$).
- Morphism computations (action on *homsets*, Velu, generalized j-invariants, characteristic polynomials of endomorphisms and norms of isogenies).
- Analytic construction of Drinfeld modules (logarithm and exponential).

User-oriented design:

- Simple, high-level, elegant interface.
- Exhaustive, useful documentation.
- Thorough testing.
- The development is still active, with contributions welcome.
- We had great feedback from the community.

First features were released in SageMath 10.0. The rest is being reviewed.

Main features

Features:

- General constructions (Drinfeld modules, morphisms, category).
- Basic computations (evaluation, rank, height, j-invariant, action on $\overline{K}$).
- Morphism computations (action on *homsets*, Velu, generalized j-invariants, characteristic polynomials of endomorphisms and norms of isogenies).
- Analytic construction of Drinfeld modules (logarithm and exponential).

User-oriented design:

- Simple, high-level, elegant interface.
- Exhaustive, useful documentation.
- Thorough testing.
- The development is still active, with contributions welcome.
- We had great feedback from the community.

First features were released in SageMath 10.0. The rest is being reviewed.

Outline of the talk

Why this project?

What is a Drinfeld module?

Focus: the crucial question of data representation

Main features

Demo

Demo

`https://xavier.caruso.ovh/notebook/drinfeld-modules`