Séminaire Lotharingien de Combinatoire **82B** (2019)
Article #100, 6 pp.

Proceedings of the 31st Conference on Formal Power Series and Algebraic Combinatorics (Ljubljana)

Supporting Data-Driven Mathematics: Database and Interface Generation

Katja Berčič*

Department of Computer Science, FAU Erlangen-Nürnberg, Germany

Abstract. Easier access to computation is turning mathematics into an at least a partially data-driven discipline. But it does not use the tools of the data trade: a majority of datasets are still shared in an ad hoc manner. We contribute an automated database setup process which builds a complete website stack, from the database schema to a web interface, from a dataset declaration.

Keywords: mathematical data, database framework, schema declaration

1 Introduction

Generating data is becoming more and more common in mathematics, but there are no easy ways to store and share it. In particular, providing a user-friendly interface to a dataset typically represents a non-trivial investment of resources for an author. Many datasets, especially the smaller ones, end up stored and shared in an ad hoc manner. Text files with one entry per line or code that produces an array of objects for a computer algebra system (and similar) are not uncommon.

Related Work The work in progress living survey of mathematical databases [2] is an indication of the status of data in mathematics. It currently lists about a hundred datasets in roughly 50 table entries. The OEIS is the oldest mathematical database, and arguably the most influential one. The LMFDB covers a wider scope and collects several datasets related via Langland's program [5]. The OEIS and the LMFDB both have a substantial mathematical knowledge management architecture, as do some other larger database projects. Several datasets have also been integrated with computer algebra systems such as GAP, Magma or SageMath. Both of these options are out of reach for smaller collaborations.

Some projects, like the LMFDB and the House of Graphs, accept dataset submissions. However, they are all limited to a specific area of mathematics, and the authors need to be registered users. Generic research data hosting services are not extensively used by mathematicians, likely due to the combination of the lack of awareness of them and the

*katja.bercic@fau.de The author was supported by EU grant Horizon 2020 ERI 676541 OpenDreamKit

M	$\mathrm{Tr}(M)$	Orthogonal	σ_M	$\det(\lambda I - M)$
$\left(\begin{smallmatrix} 2 & 0 \\ 0 & 1 \end{smallmatrix}\right)$	3	yes	2, 1	$\lambda^2 - 3\lambda + 2$
$\left(\begin{smallmatrix} 2 & 1 \\ 1 & 2 \end{smallmatrix}\right)$	4	no	3, 1	$\lambda^2 - 4\lambda + 3$
$\left(\begin{smallmatrix} -1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$	0	yes	1, -1	$\lambda^2 - 1$

Table 1: Running example: Mathilde's matrix dataset

lack of added value they provide. Some such are the EU infrastructures EUDAT and EOSC, RADAR, and hosting provided by publishers. Tools for managing and browsing general databases are similarly unused. They tend to be rich in features that are not useful from the perspective of mathematical datasets and have an initially steep learning curve. Somewhat more surprising is that even simpler database tools, e.g. those based on the light-weight SQLite database, appear to not be used much or at all.

The framework presented in this abstract is called MBGen [7] and it builds on two projects. The concepts of mathematical schemata and codecs were developed during the OpenDreamKit project [10]. The interface that is produced is essentially an instance of a DiscreteZOO website [6], which is a part of a larger project that also includes a data repository and a SageMath package. The DiscreteZOO project [4] was developed for the use case of collections of graphs with a high degree of symmetry, but was designed to be useful in a more general setting.

Contribution We automate the process of setting up a database with a web interface for mathematical datasets. MBGen takes a dataset declaration as input and generates a complete website stack, from the database schema to a web interface. The contribution is twofold and in the original spirit of DiscreteZOO. Authors of mathematical datasets will benefit from an automated process of database setup. In turn, this should lead to more datasets being available with a GUI and easy to use search features, which will benefit the researchers (data consumers and users). A standardized interface can also form a basis for cross-database sharing and linking in the future. In most cases, even a simple software stack would be a great improvement over the form a dataset is currently shared as.

Running example The dataset shown in Table 1 is simple on purpose and we will use it to demonstrate the workings of MBGen. Mathilde has collected a set of integer matrices and computed their trace, eigenvalues, whether they are orthogonal, and their characteristic polynomial. She wants to publish the dataset on her website.

2 The MBGen Framework

MBGen [7] works for datasets that fit the following simple mold. First, the dataset is composed of some number of lists of mathematical objects of the same type. Second, for every type, the list consists of either simple objects or of object records. These records could contain information about a set of mathematical properties or invariants, provided they can be reasonably represented in a database. The set of supported types is extensible and currently contains some basic types (Booleans and integers) as well as some more complex types for demonstration. An up-to-date description of the system, including the setup and a list of supported types is available in the project readme.

Writing a Dataset Declaration The schema declaration is written in the Mathematical Data Declaration Language (MDDL) and is essentially a list of declarations $c : A$, where c is a name of a property and A is its type. Every declaration also needs a codec, which is simply a pair of mappings (an encoding and a decoding) between a mathematical type and a database type. MDDL and its underlying infrastructure were introduced and described in [3]. For the purpose of this abstract, we only demonstrate how MDDL can be used through examples, and briefly describe codecs in the following paragraph.

Codecs are an essential part of the system that handles the communication between the mathematical level and the database level. They enable interoperability, which happens at the mathematical level, and connect it with the storage at the database level. Note that codecs can be arbitrary code and arbitrarily complex: consider the example Brendan McKay's graph formats [8], which are widely used to represent graphs as strings. These formats are typically used in combination with canonical labelings [1] to ensure that two graphs with the same encoding are necessarily also isomorphic.

To set up her project, Mathilde follows the steps described on the project page [7]. Once she is ready, she starts writing her schema declaration. First, she just writes down an empty theory called Matrices (Listing 1). Next, she will write property declarations for each of the columns in her dataset, as in Listing 2.

Listing 1: The empty theory

```
namespace http://data.mathhub.info/schemas
theory Matrices : ?MDDL =
```

Listing 2: A property declaration

```
property : propertyMathType |
    meta ?Codecs?codec propertyCodec |
    propertyTags |||
```

She only needs to use types that are already supported, and for which codecs have already been defined. In addition to the required information about the mathematical type and the codecs, Mathilde can use the optional tags (Table 2) to specify the interface behaviour.

The tag `hidden` can be used when a dataset has many columns (like the graph datasets in DiscreteZOO [6]). If there are hidden columns, the interface shows an additional widget, which lets a visitor choose which columns they want to see. If the tag `display`

Database	`index`	Column representing a dataset index
Frontend	`collection`	Collection metadata, e.g. provenance data
	`display`	Display name in the list of filters and in the results table head.
	`hidden`	By default, hide in the results display.
	`opaque`	Do not use for filtering.

Table 2: Metadata tags

is not present, the column name is used for display. The tags particularly relevant for the interface the metadata tags and the tags `hidden` and `opaque`.

Now, Mathilde can write down her dataset declaration (Listing 3).

Listing 3: Mathilde's dataset declaration

```
theory Matrices : ?MDDL =
meta /schemas?MDDL?datasetName "Mathilde's matrix dataset"
    mat: matrix ℤ 2 2 meta ?Codecs?codec MatrixAsArray IntIdent
        tag ?MDDL?opaque
    trace: ℤ meta ?Codecs?codec IntIdent
    orthogonal: bool meta ?Codecs?codec BoolIdent
    eigenvalues: list ℤ meta ?Codecs?codec ListAsArray IntIdent
        tag ?MDDL?opaque
    characteristic : Polynomial IntegerRing meta ?Codecs?codec PolynomialAsSparseArray IntIdent
        tag ?MDDL?opaque
```

Generating and Populating the Database For every schema theory T, MBGen generates an SQL table. The table name is the theory name and for each declaration $c : A$ in T, it generates a column, whose type is obtained from the codec declaration. In addition to these, MBGen generates a primary key column called `ID` of type `UUID`. When Mathilde runs MBGen on her schema theory from Listing 3, she obtains the database schema on the right.

Listing 4: Schema

```
     Column      |   Type
-----------------+----------
 ID              | uuid
 MAT             | integer[]
 TRACE           | integer
 ORTHOGONAL      | boolean
 EIGENVALUES     | integer[]
 CHARACTERISTIC  | integer[]

Indexes: "Matrices_pkey"
PRIMARY KEY, btree ("ID")
```

If the dataset has a pre-existing ID-like field, which is not a `UUID`, the corresponding column should be tagged with `index`. She can insert her data using the corresponding SQL `INSERT` statements, see Listing 5.

Listing 5: Table contents after insert

```
          ID          |     MAT      | TRACE | ORTHOGONAL | EIGENVALUES | CHARACTERISTIC
----------------------+--------------+-------+------------+-------------+----------------
 e278b5e8-4404-...    | {2,0,0,1}    |     3 | t          | {2,1}       | {0,2,1,-3,2,1}
 05a30ff0-4405-...    | {2,1,1,2}    |     4 | f          | {3,1}       | {0,3,1,-4,2,1}
 1be3f022-4405-...    | {-1,0,0,1}   |     0 | t          | {1,-1}      | {0,-1,2,1}
```

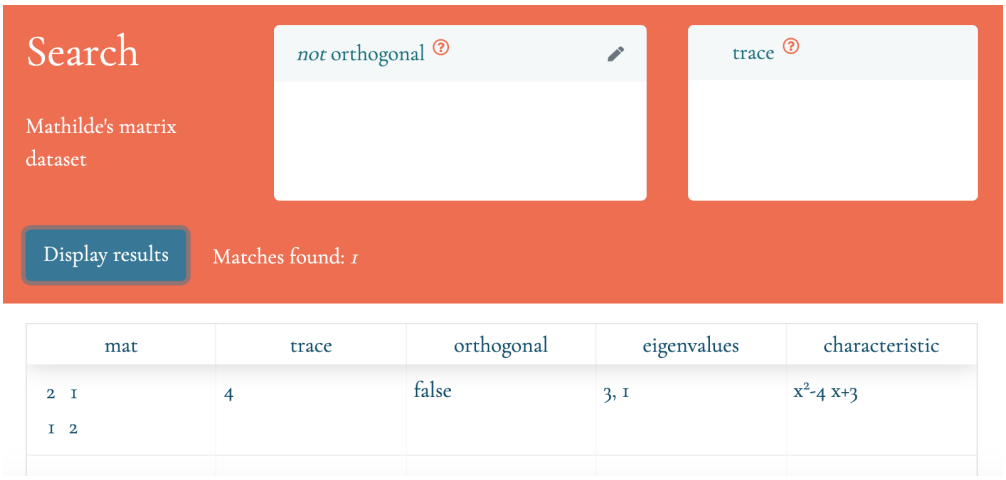

Figure 1: Screenshot of the website for Mathilde's use dataset

Extensions and Customisation Mathilde can modify the code and the database after they are generated and before running or deploying the website. We use the project wiki [7] to document how such modifications can be made.

3 Conclusion and Future Work

To evaluate the MathDataHub setup, we ran MBGen for the datasets hosted on the DiscreteZOO website. In some sense, the DiscreteZOO datasets are simpler than Mathilde's much smaller dataset, since they only contain Boolean and integer valued properties, while the objects (graphs) are string-encoded. An author of a similarly uncomplicated dataset with no new codecs could start from an existing dataset declaration and adapt it to their needs in under an hour.

MBGen will be used as a critical first step in MathDataHub, a work in progress project that has as its goal a unified mathematical data infrastructure, which will likely include hosting options. As a part of that, we will continue to expand the core library of codecs and support for common data types. Similarly, we intend to make it easier to add additional (custom) filters with future versions. We will use [9] as a starting point to explore possibilities for the development of mathematical query languages. Finally, we aim to provide an API that would be suitable for providing access from computational software such as SageMath or GAP.

Acknowledgements

The author gratefully acknowledges Dennis Müller and Tom Wiesing's help in getting to

grips with MMT, as well as Tom Wiesing's invaluable help in setting up the harmonious running of the various parts of the framework. The author would also like to thank Michael Kohlhase for several constructive discussions which were instrumental in setting the direction of the project and placing it into the larger setting. The author is supported by the EU grant Horizon 2020 ERI 676541 OpenDreamKit.

References

[1] L. Babai and E. M. Luks. "Canonical Labeling of Graphs". *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*. STOC '83. ACM, New York, NY, USA, 1983, pp. 171–183. Link.

[2] K. Berčič. *Math Databases Living Survey*. Link (visited on 03/19/2019).

[3] K. Berčič, M. Kohlhase, and F. Rabe. "Towards a Unified Mathematical Data Infrastructure: Database and Interface Generation". *Intelligent Computer Mathematics*. Conferences on Intelligent Computer Mathematics. Ed. by C. Kaliszyck, E. Brady, C. Sacerdoti-Coen, and A. Kohlhase. In preparation. 2019. Link.

[4] K. Berčič and J. Vidali. "DiscreteZOO: a Fingerprint Database of Discrete Objects". 2018. Link.

[5] S. Bernstein Joseph Gelbart, ed. *An Introduction to the Langlands Program*. Birkhäuser, 2003.

[6] *DiscreteZOO website*. Link (visited on 03/19/2019).

[7] *MBGen description, links demos and code repositories*. Link (visited on 03/19/2019).

[8] B. McKay. "Description of graph6, sparse6 and digraph6 encodings". Link.

[9] F. Rabe. "A Query Language for Formal Mathematical Libraries". *Intelligent Computer Mathematics*. Springer Berlin Heidelberg, 2012, pp. 143–158. arXiv:1204.4685.

[10] T. Wiesing, M. Kohlhase, and F. Rabe. "Virtual Theories – A Uniform Interface to Mathematical Knowledge Bases". *Mathematical Aspects of Computer and Information Sciences*. Lecture Notes in Computer Science 10693. Springer, Cham, 2017, pp. 243–257. Link.