



Disclaimers

Since it is an itch of mine, will mostly talk about how we might use alignments to make data more useful.



Hopefully, the current (small) momentum behind data might give some energy to the alignments project.

1. Overview

A trilogy in four parts

2. A data safari

Survey with an eye towards concepts

3. Aligning concepts?

Data management, a TLDR

4. Aligning objects

An example (more like item 3a)

5. Data and formalization

Alignments?

Last slide: QR code linking to slides and a list of references.



1. Overview

A trilogy in four parts

2. A data safari

Survey with an eye towards concepts

3. Aligning concepts?

Data management, a TLDR

4. Aligning objects

An example (more like item 3a)

5. Data and formalization

Alignments?

Last slide: QR code linking to slides and a list of references.

A long time ago, when I was a young and naive undergraduate student, I built a database of tournaments for the Slovenian Go Association. Soon after that I started my PhD and one of the first things I encountered was my advisor's dataset of cubic vertex-transitive graphs. To me, it seemed rather urgent that something like that belonged in a database, not just as a plain text file.

- To start, we'll spend some time on a safari of mathematical data.
- We will look at a use case for alignments: annotating data, making it more useful.
- We will (perhaps cheekily) stretch the notion of alignments to objects.
- Finally, we will consider an example of incorporating data with formalization.

If we're counting generously, this slide could be the "fifth part," making this talk a trilogy in five parts. Don't panic; there won't be a sixth.

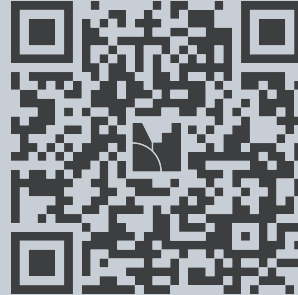
Warming up with a poll

Thank you for being patient with me

Scan the QR code on the right
or go to menti.com and enter
the code **3882 2786**

When answering, consider

- who produces the data,
- who are the users,
- what is the content.



2. A data safari

Remembering that the universe is big

- "Research data are all digital and analog objects generated or handled in the process of doing research" (*MaRDI*)
 - *Right*: the listing on the Cornell Mathematics Library page on "Math Databases".
- MathSciNet
 - Zentralblatt für Mathematik
 - Google Scholar
 - Wikipedia
 - MacTutor History of Mathematics
 - Scopus
 - The Web of Science
 - Mathworld
 - Jahrbuch-Project Electronic Research Archive for Mathematics (mathematics literature 1868 - 1943)
 - arXiv
 - ERIC (index in the field of Education, including Education in Mathematics)
 - Wolfram|Alpha ("allows you to enter a query and returns an answer from structured data")

"All digital and analog objects" includes: paper publications, proofs, computational results (and more). Does this mean that all mathematicians should have a research data management plan when they start writing a paper? Probably not, but perhaps they should.

Modern mathematical databases

Four well-established examples, just in case

The On-Line Encyclopedia of Integer Sequences (OEIS) is a database of integer sequences. It is founded in 1964 by N. J. A. Sloane. The search bar shows the sequence A000045 (Fibonacci numbers). The LMFDB (The L-functions and modular forms database) is a database of L-functions and modular forms. It includes sections for A database, Announcements, Learn more, and Citations and acknowledgments. The House of Graphs is a database of graphs. It shows a search for 20 graphs found that satisfy the following conditions: 20 most recently added graphs. The table lists graphs with columns: HOG ID, Name, Number of Vertices, Number of Edges, Minimum Degree, and Maximum Degree. The pi-Base is a community database of topological counterexamples. It includes a search bar and a list of counterexamples.

The On-Line Encyclopedia of Integer Sequences (OEIS) is a database of integer sequences. It is founded in 1964 by N. J. A. Sloane. The search bar shows the sequence A000045 (Fibonacci numbers). The LMFDB (The L-functions and modular forms database) is a database of L-functions and modular forms. It includes sections for A database, Announcements, Learn more, and Citations and acknowledgments. The House of Graphs is a database of graphs. It shows a search for 20 graphs found that satisfy the following conditions: 20 most recently added graphs. The table lists graphs with columns: HOG ID, Name, Number of Vertices, Number of Edges, Minimum Degree, and Maximum Degree. The pi-Base is a community database of topological counterexamples. It includes a search bar and a list of counterexamples.

HOG ID	Name	Number of Vertices	Number of Edges	Minimum Degree	Maximum Degree
53687	Asymmetric graph (3322710)	7	6	0	3

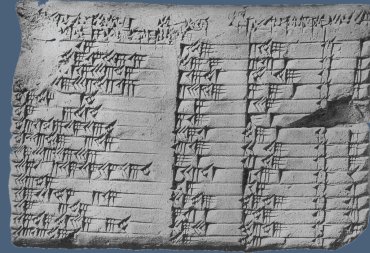
I mostly think about data that broadly looks like the databases above (possibly with less infrastructure): collections of (counter)examples

Some in audience have probably seen OEIS, LMFDB. Most datasets are not so complex (in the infrastructure sense).

Cuneiform and the Greeks

Data probably goes as far back as math

- Earliest datable table containing mathematical computations: length measurements, corresponding areas (Sumer, c. 2600 BCE).
- *picture*
A list of Pythagorean triples (Babylon, c. 1800 BCE).



Plimpton 322 clay tablet; Bill Casselman

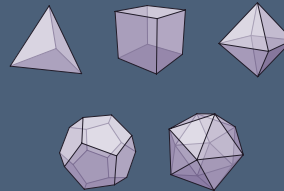
Eleanor Robson, "Tables and tabular formatting in Sumer, Babylonia, and Assyria, 2500 BCE-50," Campbell-Kelly et al [eds]. The History of Mathematical Tables from Sumer to Spreadsheets [2003]

Cuneiform and the Greeks

Data probably goes as far back as math

- Earliest datable table containing mathematical computations: length measurements, corresponding areas (Sumer, c. 2600 BCE).

- *picture*
The list of the Platonic solids (a.k.a. convex regular polyhedra; *Theaetetus* of Athens, c. 417 - c. 369 BCE).



Platonic solids; Wikipedia user [Drummyfish](#)

Platonic solids: constructed by regular polygons as faces, with the same number of them at each vertex, convex.

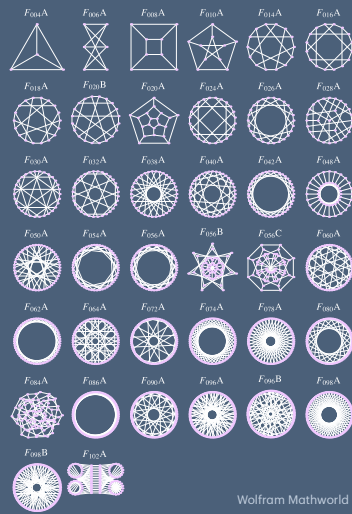
- [illegible]

The Foster Census

Begun in 1930, published in 1988 as a book.

Cubic symmetric graphs with
up to 512 vertices.

There are 207 such graphs.
Foster only missed a handful.



Wolfram Mathworld

Symmetric graph: every ordered pair of adjacent vertices (an arc) can be mapped to any other such pair.

Indexing mathematical data

Is surprisingly hard

1. Provide a searchable index of existing mathematical databases.
2. Support mathematicians in creating new mathematical databases and improving existing ones.

MATHBASES

INDEX OF MATHEMATICAL DATABASES

Info	Name
	(S-)Unit Groups of Orders ↗ Gabriele Nebe, Renaud Coulanges, Oliver Braun, Sebastian Schönnenbeck
	A Catalogue of Lattices ↗ Neil Sloane, Gabriele Nebe
	A Census of Edge-Transitive Tetravalent Graphs ↗ Primož Potočnik, Steve Wilson
	A Database for Number Fields ↗ Jürgen Klüners, Gunter Malle
	A Database of Continued Fractions of Polynomial Type ↗ Henri Cohen
	A Database of Galois polynomials ↗ Bill Allombert, Igor Schein
	A Database of Graphs in Combinatorica Format ↗ Sriram Pemmaraju, Steven Skiena
	Adjectives Project ↗ Jesse Vogel, David Holmes
	An Atlas of Abstract Regular Polytopes for Small Almost Simple Groups ↗ Laurence Vauthier, Dimitri Leemans
	An Atlas of Chiral Polytopes for Small Almost Simple Groups ↗ Dimitri Leemans, Michael Hartley, Isabel Hubard

MathBases began as an effort to map what is out there as well as to index and showcase mathematical databases. There is a bias towards number theory and combinatorics, which reflects the editors' familiarity with these areas.

MathBases has gained surprising visibility, we could promote use of standardized vocabulary.

The diversity of math databases

Four easy dimensions

small - large

Atlas of Small Chiral Polytopes (56)

Lists of finite lattices ($17 \cdot 10^9$)

enumerations and classifications - curated/collected

The Foster census

OEIS, House of Graphs

stored - generated on demand

House of Graphs - The Small Groups Library in GAP - nauty

easy to obtain values - single value hard to obtain

nauty

Parameters of Strongly Regular Graphs

Despite the restriction we have imposed, there is great diversity.

The diversity of math databases

Another dimension - purpose?

- Curating (counter)examples
- Index (fingerprint) theorems via simpler objects
- Knowledge reference
- Concept instantiation
- Benchmarking



- Curation of examples: topological spaces, properties and theorems in π -base, graphs and invariants in the House of Graphs.
- Index theorems: integer sequences (OEIS), Parameters of Strongly Regular Graphs.
- Knowledge reference: definitions and properties of special functions in DLMF
- Instantiation: datasets in algebraic geometry (only one object, variety)
- Benchmarking: SuitSparse matrix dataset

The larger databases are pretty good at defining their concepts.

Concepts in databases

Large databases usually define relevant concepts

Properties

Filter

Id	Name	Description
P1	T₀ Kolmogorov, T0	Given any two distinct points, there is an open set containing one but not the other.
P2	T₁ Fréchet, T1	Given any two distinct points, each has a neighborhood not containing the other point.
P3	T₂ Hausdorff, T2	Given any two distinct points a and b , there are disjoint open sets O_a and O_b containing a and b , ...
P4	T_{2½} Urysohn, Completely Hausdorff, T2.5	Distinct points are separated by closed neighborhoods. In other words, given any two distinct points...
P5	T₃ Regular Hausdorff, T3	A space which is both Regular and T₂ . Equivalently, a space that is both Regular and T₀ .
P6	T_{3½} Tychonoff, Completely regular Hausdorff, T3.5	A space which is both Completely regular and T₂ .

Concepts in databases

Large databases usually define relevant concepts



The House of Graphs

[Search](#)

[Draw Graph](#)

[Meta-directory](#)

[Publications](#)

[Help](#)

[Log in](#)

[Register](#)

Help

Frequently Asked Questions

Which graph formats are supported by House of Graphs?



Which invariants are supported by House of Graphs?



Following invariants are currently supported.

Acyclic

A graph is said to be *acyclic* if it contains no cycle.

Algebraic Connectivity

The *algebraic connectivity* of a graph G is the second smallest eigenvalue of its Laplacian matrix L , defined by $L = D - A$ where A is the adjacency matrix of G and D is a diagonal matrix where D_{ii} is the degree of vertex i .

Average Degree

The *average degree* of a graph G is the (arithmetic) mean of the degrees of its vertices.

Bipartite

A graph G is *bipartite* if it is possible to partition the set of vertices of G into two subsets A and B such that every edge of G joins a vertex of A to a vertex of B .

Chromatic Index

The (edge) *chromatic index* of a graph G is the minimum number of different colors required to color the edges of G such that two edges incident to the same vertex have different colors.

Chromatic Number

The *chromatic number* of a graph G is the minimum number of different colors required to color the vertices of G such that two adjacent vertices have different colors.

Circumference

The *circumference* of a graph is the order of the largest subgraph of the graph that is a cyclic graph.

Claw-Free

A graph is *Claw-Free* if it does not contain a $K_{1,3}$ as an induced subgraph.

Clique Number

The *clique number* of a graph G is the maximum cardinality of a clique in G .

Microsoft

Where is the (rest of the) data?

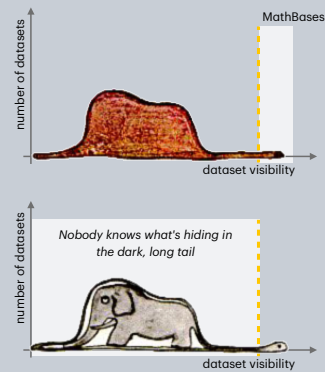
We are still looking for the long tail

- An analysis of zbMATH references revealed a long tail of data

(Hulek, Müller, Schubotz and Teschke, Mathematical Research Data)

- *Call to action*

What are relevant metadata for collections of examples?



There are currently 128 entries in MathBases, a few more waiting to be entered. Is this all?

What you see on MathBases is probably just the tip of the iceberg (even if we're trying to make this not the case).

Questions?

3. Aligning concepts

Data provide a use case; buzzwords

Making data more useful requires aligning the data with the right concepts

- FAIR: Findable, Accessible, Interoperable and Reusable.
Not the same as open data
(free use, accessibility, submission)
- RDMP: Research Data Management plan



I want to illustrate my frustration with some of the data in mathematics.

Let us consider the specimen of industrial design pictured on the right.

There are red arrows on the right and blue arrows on the left. The main handle turns counter-clockwise and seemed to correlate somewhat with water output. The smaller handle appeared to have little effect. I messed around with it for a while and could not get the water to be warmer than lukewarm.

As a last resort, I turned the handle all the way into the worryingly blue zone, which unexpectedly but fortunately resulted in hot water.

3. Aligning concepts

Data provide a use case; buzzwords

Making data more useful
requires aligning the data with
the right concepts

- FAIR: Findable, Accessible,
Interoperable and Reusable.
Not the same as open data
(free use, accessibility, submission)
- RDMP: Research Data
Management plan

List of the 17 representatives of
 $IC(6,3)$, ordered by the RevLex-Index.
The numbers above the signs indicate
the elements of the corresponding basis.

```
11121121231121231234
22332334442334445555
34445555555666666666

IC(6,3, 1) = ++++++---++
IC(6,3, 2) = ++++++---+
IC(6,3, 3) = ++++++---+
IC(6,3, 4) = ++++++---+
IC(6,3, 5) = 0+++++---+
IC(6,3, 6) = 0+++++---+
IC(6,3, 7) = 0+++++---+
IC(6,3, 8) = 0+++++---+
IC(6,3, 9) = 0+++++---+
IC(6,3,10) = 0+++++---+
IC(6,3,11) = 0+++++---+
IC(6,3,12) = 0+++++---+
IC(6,3,13) = 0+++++---+
IC(6,3,14) = 0+++++---+
IC(6,3,15) = 0000+++++
IC(6,3,16) = 0000+++++
IC(6,3,17) = 00000000+++++
```

The first time I saw the data on the right side of the slide, I had no idea what I was looking at. As with the tap, it is possible to figure it out (I presume), given enough time.

The FAIR guiding principles were published in 2016 and are an attempt to describe, in very general terms, how usable data look like. Importantly some of what they recommend is related to alignments.

Open in "open access" refers to the removal of financial, legal and technical barriers to data, while accessibility in FAIR refers to the data being retrievable by humans and machines.

The RDMP is another buzzword that appears in relation to research data. We won't spend too much time on it, but I thought it would be helpful to at least mention it.

The FAIR guidelines

Mostly about metadata, deep FAIR: proposed term for corresponding properties for objects

Findable

globally unique, persistent IDs,
rich metadata, indexing
*easy to identify and find for both humans
and computers, e.g. with metadata that
facilitate searching for specific datasets*

Accessible

long term storage, well-
defined access conditions,
(metadata or data level)

Interoperable

FAIR **knowledge
representation language**
*ready to be combined with other datasets
by humans or computers, without
ambiguities in terms and values*

Reusable: clear usage

license, provenance, domain-
relevant standards,
**comprehensive, relevant
attributes**
*ready to be used for future research and to
be further processed using computational
methods*

The FAIR guidelines are intentionally broad and somewhat vague; designed to provide communities with a flexible framework to be further developed and adapted. They focus primarily on metadata — covering aspects such as authorship, provenance, licensing, and descriptions of the dataset's contents.

Adopting FAIR principles can significantly improve the citability, visibility, and confirmability of datasets, making computational results more easily reproducible.

In particular, the principles stress the importance of accurately and understandably describing the data so that it is findable, interoperable, and reusable.

A brief note on interoperability: to the best of my knowledge, there is currently no standard knowledge representation language for mathematical data. While this means we don't yet have to worry about strict interoperability requirements, it is an area where the community should invest effort in the future.

An aside: the RDMP

Research Data Management Plan

- A living document, from the start of a project.
- Outlines how data will be managed throughout a research project.
- Incorporates the FAIR principles to ensure that data is handled in a way that maximizes its long-term value and usability.

Data Management Plan

No data management plan is necessary, since the research outlined in this proposal is in the realm of Mathematics and by nature theoretical. The PI will make his research freely and publicly accessible through articles, graphics and programs on the web page <INSERT WEBSITE>. The PI will also submit articles and papers to appropriate peer-reviewed journals for publication. Finally this work will be disseminated through academic research seminars and conferences.

www.math.harvard.edu/media/DataManagement.pdf
(link courtesy of Boege et al)

The other buzzword you might encounter is the RDMP, a document outlining the plan for managing the data. It can incorporate the FAIR principles and can be required by funding agencies and/or institutions.

I'm primarily bringing it up because of the example on the right, which echoes a claim I've often heard in and about the field of mathematics. It is that mathematicians rarely produce data, and that the data they do produce requires little to no management. I've also frequently come across statements like "you can't license mathematical objects" and the belief that if data is posted on someone's website, it is automatically in the public domain and freely usable.

This attitude is an obstacle (of a social nature) that stands in the way of better data in math.

Take-aways for data management

At this point in time and for people compiling datasets containing examples

- **More and better metadata and documentation**

waiting for a metadata standard

- The meaning and provenance of mathematical data can require complex mathematical data.

- *Archiving and preservation:* snapshots in a machine readable format on Zenodo or GitHub to ensure longevity.

- *Reproducibility* for results of computation: record software info (version), attach code.

- Open problem: recognition beyond publications*

There are a few things dataset authors can do to improve the usability, and documenting the contents and structure is an important part of that.

You might remember that one of the stated goals of MathBases is to help mathematicians compile new databases and improve existing ones. Because it has gained some visibility (which came as a little bit of a surprise to me), we are in a position to recommend good practices. What should we recommend?

A generic database

Not the data model, but good enough for this talk



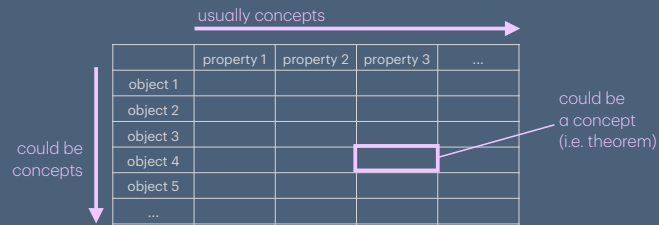
Values

- results of computations: bipartiteness, chromatic number, ...
- curated entries: theorems, references, code, equations, ...

For now, let's take a simple table as a model for datasets - it is going to be good enough.

A generic database - aligning with concepts

Not the data model, but good enough for this talk



Values

- results of computations: bipartiteness, chromatic number, ...
- curated entries: theorems, references, code, equations, ...

Ideally, all of these should be nicely annotated: properties, but also some of the objects and possibly some of the object-property values.

How can we annotate the data? Few options, none great • Write down all the definitions (a lot of work) • Link to Mathlib or similar (a moving target, not so friendly for such a use) • Link to Wikipedia or similar Note: for usability, anything is better than nothing.	Girth The girth of G is the length of the shortest cycle (if any) in G.	Group Size The group size of a graph G is the size of the automorphism group of G, i.e. $\text{Aut}(G)$.	Hamiltonian A graph is Hamiltonian if it contains a Hamiltonian cycle. A Hamiltonian cycle is a cycle which contains every vertex of the graph.
	Hypohamiltonian A graph G is hypohamiltonian if G is non-hamiltonian and $G-v$ is hamiltonian for every vertex v of G.	Hypotractable A graph G is hypotractable if G is non-tractable and $G-v$ is tractable for every vertex v of G.	Independence Number The independence number of a graph G is the maximum cardinality of an independent set in G.
	Index The index or spectral radius of a graph is the greatest eigenvalue of its adjacency matrix.	Laplacian Largest Eigenvalue The Laplacian largest eigenvalue of a graph G is the largest eigenvalue of its Laplacian matrix L, defined by $L = D - A$ where A is the adjacency matrix of G and D is a diagonal matrix where D_{ii} is the degree of vertex i.	Length of Longest Induced Cycle The order of the largest induced subgraph of the graph that is a cycle.
	Smallest Eigenvalue The smallest eigenvalue of a graph is the smallest eigenvalue of its adjacency matrix.	Traceable A graph is traceable if it contains a Hamiltonian path. A Hamiltonian path is a path which contains every vertex of the graph.	Treewidth Tree treewidth of a graph G is the minimum width among all possible tree decompositions of G. A tree decomposition is a tree T such that every vertex x in $V(G)$ has an associated bag $B(x)$ that is a subset of $V(G)$. Every edge e in $E(G)$ must be a subset of at least one bag $B(x)$. Moreover, for every vertex v in $V(G)$, the set of tree vertices whose bags contain v induce a

Let's look at defining one of the properties.

Suppose we take a dataset containing graphs; one of the properties could be whether a graph is Hamiltonian.

Big databases usually have the contributors to do the work of writing down all the definitions.

For smaller projects, property definitions are usually not defined (except where there is potential for ambiguity) - they are expected to be known by the members of the community, the main users. Unfortunately, this makes the data less useful outside of the area of origin.

How can we annotate the data?

Few options, none great

- Write down all the definitions (a lot of work)
- Link to Mathlib or similar (a moving target, not so friendly for such a use)
- Link to Wikipedia or similar

Note: for usability, anything is better than nothing.

Mathlib.Combinatorics.SimpleGraph.Hamiltonian

Hamiltonian Graphs

In this file we introduce Hamiltonian paths, cycles and graphs.

Main definitions

- `SimpleGraph.Walk.IsHamiltonian`: Predicate for a walk to be Hamiltonian.
- `SimpleGraph.Walk.IsHamiltonianCycle`: Predicate for a walk to be a Hamiltonian cycle.
- `SimpleGraph.IsHamiltonian`: Predicate for a graph to be Hamiltonian.

```
def SimpleGraph.Walk.IsHamiltonian
  {α : Type u.1} [DecidableEq α] {G : SimpleGraph α}
  {a b : α} (p : G.Walk a b) :
  Prop
```

A Hamiltonian path is a walk `p` that visits every vertex exactly once. Note that while this definition doesn't contain that `p` is a path, `p.isPath` gives that.

► Equations

```
theorem SimpleGraph.Walk.IsHamiltonian.map
  {α : Type u.1} {β : Type u.2} [DecidableEq α]
  [DecidableEq β] {G : SimpleGraph α} {a b : α}
  {p : G.Walk a b} {H : SimpleGraph β} (f : G →g H)
  (hf : Function.Bijective hf) (hp : p.isHamiltonian) :
  (Walk.map f p).isHamiltonian
```

```
def SimpleGraph.IsHamiltonian
  {α : Type u.1} [DecidableEq α] [Fintype α]
  (G : SimpleGraph α) :
  Prop
```

A Hamiltonian graph is a graph that contains a Hamiltonian cycle.

By convention, the singleton graph is considered to be Hamiltonian.

► Equations

```
theorem SimpleGraph.IsHamiltonian.mono
```

We could recommend linking to one of the formal mathematics libraries; not super readable, potentially too precise.

About two years ago I asked my office mate at the time to mark up definitions in the Agda Unimath library with references to other resources. It took repeated convincing that it is ok for these references to have subtle differences in the definition of the concept before the annotations took off. I suspect part of it was that linking to accessible resources made it easier for undergraduate students to join the formalization project. Linking to a formal library from a database goes in the other direction but might present a similar (or worse) problem.

How can we annotate the data?

Few options, none great

- Write down all the definitions (a lot of work)
- Link to Mathlib or similar (a moving target, not so friendly for such a use)
- Link to Wikipedia or similar

Note: for usability, anything is better than nothing.

Hamiltonian

13 languages

Article

Talk

Tools

From Wikipedia, the free encyclopedia

シムリ
ハミルトン

Look up [Hamiltonian](#) in Wiktionary, the free dictionary.

Hamiltonian may refer to:

- Hamiltonian mechanics, a formalism based on:
 - Hamiltonian (mechanics), a function that represents the total energy of a system
- Hamiltonian (quantum mechanics), an operator corresponding to the total energy of that system
 - Dyall Hamiltonian, a modified Hamiltonian with two-electron nature
 - Molecular Hamiltonian, the Hamiltonian operator representing the energy of the electrons and nuclei in a molecule
- Hamiltonian (control theory), a function used to solve a problem of optimal control for a dynamical system
- Hamiltonian path, a path in a graph that visits each vertex exactly once
- Hamiltonian matrix, a matrix with certain special properties commonly used in linear algebra
- Hamiltonian group, a non-abelian group the subgroups of which are all normal
- Hamiltonian economic program, the economic policies advocated by Alexander Hamilton, the first United States Secretary of the Treasury

See also [edit]

An experiment: Mathswitch

Lessons learned

- Wikipedia and Wikidata imperfect (not news).
- Very hard to convince practitioners to link to concepts that are slightly different.
- Hypothesis: alignments are primarily a social engineering (technology might help, but won't solve)

mathswitch — infrastructure for relaying and exchanging mathematical concepts

Search for a concept

Examples: circuit, mathematical group, Stone duality, normal

About

Mathswitch collects entries from the following sources:

- Wikidata (11482 entries)
- Wikipedia (EN) (10088 entries)
- nLab (2911 entries)
- Mathworld (6386 entries)
- ProofWiki (1840 entries),
- Encyclopedia of mathematics (938 entries),
- Agda Unimath (**432 entries**) - **835 entries**

The entries are organized into a concept network that connects the same concept appearing in different sources.

mathswitch — infrastructure for relaying and exchanging mathematical concepts

circuit

graph theory object

MathWorld: Circuit
Wikipedia (English): Cycle_(graph_theory)
Agda Unimath: circuit
Wikidata: circuit
Agda Unimath: closed walk (undirected graph)
Wikidata: cycle
Wikidata: cycle
Agda Unimath: cycle (undirected graph)

An experiment grew out of this frustration after the Dagstuhl workshop two years ago.

- Ask libraries to link to **anything**; link in place, close to the concepts (for maintenance reasons).
- Collect links and organize them into a network to provide value.
- That alone seemed to be enough to talk friends into it, but not enough to convince Mathlib, for example.

1000+ theorems

Successor to Freek Wiedijk's 100 theorems

- 100 theorems (precursor): showcasing formalizations by keeping track of formalizations of the hundred greatest theorems (*a fixed list*)
- Indexing formalizations of a much longer (changing) list of theorems

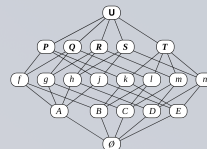
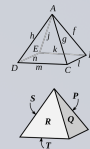
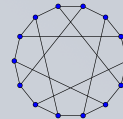
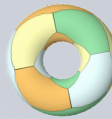
Formalized theorems								
100 entries per page			Search:					
MSC	Name	↑	Isabelle	HOL Light	Rocq	Lean	Metamath	Mizar
26	Abel's theorem						L	
12	Abel-Ruffini theorem						L	
68	Akra-Bazzi theorem						L	
11	Apéry's theorem					X		
91	Arrow's impossibility theorem					X		
16	Artin–Wedderburn theorem						L	
46	Arzelà–Ascoli theorem						L	

[1000-plus.github.io](#)

3. Aligning objects

An example: symmetric objects

- maps and hypermaps
(regular and chiral)
- graphs
(group actions transitive on vertices, edges, or arcs)
- abstract polytopes
(regular and chiral)
- maniplexes.



1. Regular map $\{6,3\}_{\text{tor}}$ on the torus ([PSL27](#))
2. Heawood graph (Koko80)
3. Hasse diagram of the square pyramid (David Eppstein)

Context: finite (combinatorial), symmetric (~transitive automorphism group)

Regular map: a decomposition of a two-dimensional manifold (such as a sphere, torus, or real projective plane) into topological disks such that every flag (an incident vertex-edge-face triple) can be transformed into any other flag by a symmetry of the decomposition.

Abstract polytope: partially ordered set which captures the dyadic property of a traditional polytope without specifying purely geometric properties such as points and lines. (1) It has just one least face and one greatest face. (2) All flags contain the same number of faces. (3) It is strongly connected. (4) If the ranks of two faces $a > b$ differ by 2, then there are exactly 2 faces that lie strictly between a and b .

- Diamond condition: for $i = 0, \dots, n - 1$ there is exactly one flag that differs from a given flag in the i -face

- Rank: $\{-1, \dots, n\}$; 0, 1 and $n - 1$ are vertices, edges and facets

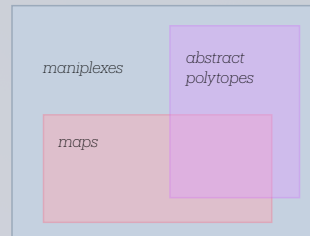
-

Top left: 16 faces, 32 vertices and 48 edges, with a symmetry group of order 192

3. Aligning objects

An example: symmetric objects

- maps and hypermaps
(regular and chiral)
- graphs
(group actions transitive on vertices, edges, or arcs)
- abstract polytopes
(regular and chiral)
- maniplexes.



Maps and abstract polytopes: group representation (Coxeter generators), Schläfli symbol;

Chiral: rotational, but no reflectional symmetry

Goal: link objects in the intersection, but also record other relations, such as maps and their skeleton graphs.

Maniplexes

A generalization of maps and polytopes

- Can be represented as edge-colored graphs
- An i -colored edge represents moving to an adjacent flag by changing the i -face.
- s_i - the transposition corresponding to swapping the vertices along i -edges
- i -faces are cycles of alternating colors

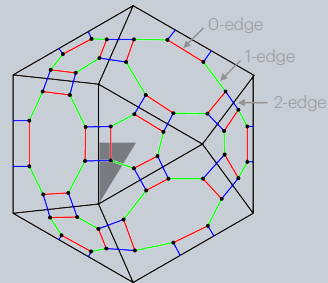


Diagram from Cunningham, Del Rio-Francos, Hubbard, Toledo, Symmetry Type Graphs of Abstract Polytopes and Maniplexes

Representations

Graph canonical labellings to the rescue

- Permutation representation
- Coxeter generator relations
- Checking isomorphism hard
- Graphs: canonical labelling
(label vertices so that
checking isomorphism
becomes checking equality)
- Use maniplexes to relate
objects

```
s0 := (4,5);;  
s1 := (3,4);;  
s2 := (2,3);;  
s3 := (1,2)(6,7);;  
poly := Group([s0,s1,s2,s3]);;
```

```
Regular 3-polytope with group of  
order 24   SD   Type [ 3, 3 ]  
[ A.1^2, A.2^2, A.3^2, (A.1 * A.3)^2,  
  (A.2 * A.1)^3, (A.3 * A.2)^3 ]
```

SD = self dual

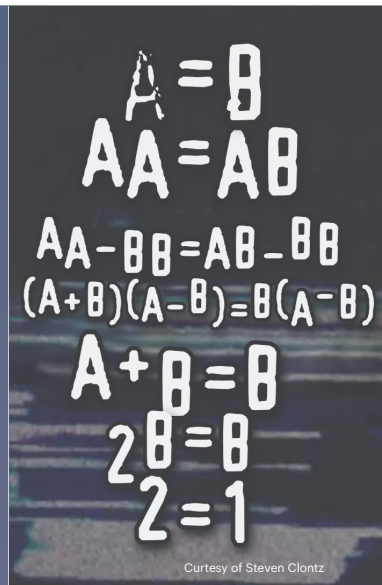
Questions?

5. Formalizing data

No answers, just an example: Lean-HoG

Formalizing a property to verify the data aligns the formalization with a column

- Source of examples for formalized mathematics
- Increased trust in the data



Is a list of examples complete and correct?

Is the connection between theory and code sound?

Are the computational results correct

In the LMFDB, they run redundant computations, integrity checks, and have found bugs in computer algebra systems.

Some standard options to increase the level of trust

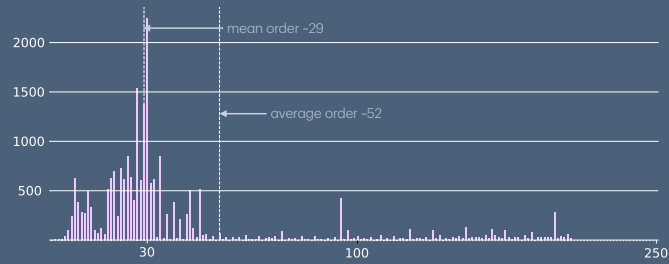
Techniques from engineering and mathematics

- Standard checks: format, type, consistency, uniqueness, ...
- *Testing*: software is run on a collection of test cases, the results are compared to reference results known to be true.
- *Redundancy*: several versions of software performing the same task are developed and executed independently, their results compared.
- Correctness of code or data is established by formal proofs.

The House of Graphs

The database of interesting graphs

- ~**32 000** graphs on up to 250 vertices,
- ~**50** properties, including computationally difficult ones, such as: genus, chromatic number, and Hamiltonicity.



The combination of graph sizes and properties means that we can't just compute whichever way we want.

Even though we formalized only a few properties, it was harder than expected.

Design options

based on quantity and complexity of objects and properties

- Prove the properties of each example by hand.
- Implement algorithm(s) in the proof assistant
(in the extreme case, implement a computer algebra system in a proof assistant)
- Encode as SAT, verify encoding to be correct, use a (trusted) solver, check the certificates provided by the solver.
- Use external software to compute properties *and their certificates*, use the proof assistant to check correctness.

1. For few objects and properties, simple.
2. Few properties, many objects, efficiently computable: can be difficult.
3. We used a combination of the last two.

Certificates (a.k.a. witnesses)

Instead of computing values, just check correctness

- Standard technique in computer science
- Check that 112909084933 is not a prime vs.
 $132241 \cdot 853813 = 112909084933$.

✓ Certificates *explain* why a property holds.

✗ Some properties do not have a certificate.

This works more broadly than you (might) think! (Used the idea for computation of election results).

The proof assistant can check the correctness of the certificate. While the connection with the property "not prime" follows directly from the definition here, this is not the case in general; a further proof that the property follows from the certificate can be necessary.



"The Petersen graph is a remarkable configuration that serves as a counterexample to many optimistic predictions about what might be true for graphs in general."

Donald Knuth

Let's look at a random example of a graph from the House of Graphs.

The Petersen graph

has a Hamiltonian path, is not Hamiltonian



A foreshadowing of things to come.

Easy-ish: find a path in the graph that visits all vertices exactly once,
Harder: prove that we can't find such a cycle in the graph.

The Petersen graph is also the smallest vertex-transitive graph that is not a Cayley graph.

Lean-HoG

A Lean 4 library for finite simple graphs incorporating the House of Graphs

- Import graphs with efficient representations into Lean,
- together with values and certificates for:
the number of connected components, bipartiteness, traceability.
- A tactic to search the database and
- a tactic to close a goal by finding an example.
- Checking the number of connected components on (almost) all graphs takes ~16h.

Mathlib provides a basic, general-purpose formalization of simple graphs, but it was not suitable for our purposes. To address this, we implemented a small library for finite simple graphs, prioritizing efficiency over generality.

Early experiments showed that we could process a graph in time at most quadratic in the number of edges, and wherever possible, sub-quadratic in the number of vertices. Working naively with lists of vertices and edges — or with adjacency matrices — led almost immediately to quadratic (or worse) time complexity.

Some invariants, such as the number of edges, can be computed efficiently by the Lean kernel, provided an efficient graph representation. For other invariants — for example, testing bipartiteness via 2-coloring or detecting odd cycles — Lean can efficiently verify a certificate when supplied.

For the invariants (traceability), with certificates that only work in one direction, one strategy would be to complement them with heuristics wherever they work. For instance, detecting a disconnected graph is an easy way to rule out Hamiltonicity. Only when these simpler methods fail would we resort to SAT solving. However, we chose to take a more principled approach by using SAT for both directions.

Warning, implementation
details ahead.

Getting graphs into Lean

Mathlib: graphs represented with a symmetric, irreflexive adjacency relation

- Given a coloring c , check that adjacent vertices have different colors:

$$\forall i j : \text{Fin } n. \text{Adj } i j \rightarrow (c i \neq c j),$$

time complexity $\mathcal{O}(n^2)$, only $\mathcal{O}(|E|)$ when given a set of edges.

- Check whether a graph is regular:

$$\exists k : \mathbb{N}. \forall i : \text{Fin } n. |\{j : \text{Fin } n; \text{Adj } i j\}| = k$$

time complexity $\mathcal{O}(n^2)$, only $\mathcal{O}(n)$ when given a neighborhood map.

Lean-HoG graph representations

A Lean 4 library for finite simple graphs incorporating the House of Graphs

All properties require efficient

- membership checking, and
- checking that something holds for every element of a set, i.e. vertices, edges.

Lean-HoG:

- RBSets and RBMaps for all sets and maps,
- graphs represented via sets of edges (and an auxiliary neighborhood map, checked to be equivalent).

Certificates:

- could use regular certificates (no SAT) for the simple direction; use heuristics whenever they work for the other direction (such as a disconnected graph for Hamiltonicity), only resort to SAT when all else fails;
- we took the more principled approach with SAT.

Connected components

```
-- Verices u and v are connected if they are related by the equivalence
-- relation generated by the adjacency relation. -/
def Graph.connected {G : Graph} : G.vertex → G.vertex → Prop := EqvGen G.adjacent

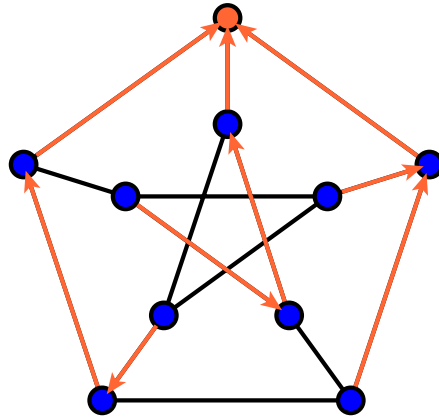
-- Connected components of a graph, as a structure -/
class ConnectedComponents (G : Graph) : Type :=
  -- Number of connected components -/
  val : Nat

  -- The component of the given vertex -/
  component : G.vertex → Fin val

  -- Components are inhabited -/
  componentInhabited : ∀ (i : Fin val), ∃ u, component u = i

  -- The assignment of components coincides with connectedness -/
  correct : ∀ u v, component u = component v ↔ G.connected u v
```

Certificate



Certificate

```
/-- A certificate for connected components -/
class ConnectedComponentsCertificate (G : Graph) : Type :=
  -- Data
  val : Nat
  component : G.vertex → Fin val
  root : Fin val → G.vertex
  next : G.vertex → G.vertex
  distToRoot : G.vertex → Nat

  -- Properties
  componentEdge : G.edgeSet.all (fun e => component (G.fst e) = component (G.snd e)) =
  rootCorrect : ∀ i, component (root i) = i
  distRootZero : ∀ (i : Fin val), distToRoot (root i) = 0
  distZeroRoot : ∀ (v : G.vertex), distToRoot v = 0 → v = root (component v)
  nextRoot : ∀ i, next (root i) = root i
  nextAdjacent : ∀ v, 0 < distToRoot v → G.adjacent v (next v)
  distNext : ∀ v, 0 < distToRoot v → distToRoot (next v) < distToRoot v

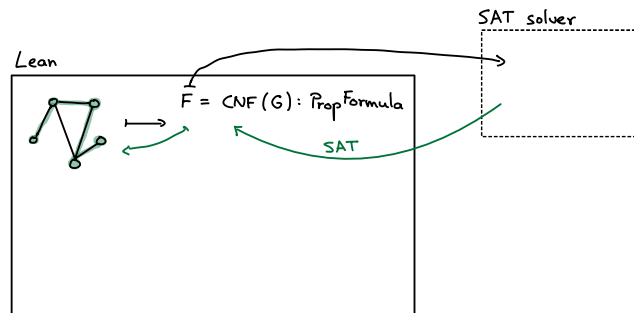
/-- From a components certificate we can derive the connected components -/
instance {G : Graph} [C : ConnectedComponentsCertificate G] : ConnectedComponents G :=
```

Load the certificate

```
--  
| JSON representation of connected components certificate.  
-/  
structure ConnectedComponentsData : Type where  
  val : Nat  
  component : Array (Nat × Nat)  
  root : Array (Nat × Nat)  
  next : Array (Nat × Nat)  
  distToRoot : Array (Nat × Nat)  
  
deriving Lean.FromJson  
  
-- Build a connected components certificate expression from the data. -/  
def buildCert (G : Q(Graph)) : ConnectedComponentsData → Q(ConnectedComponentsCertificate $G) :=
```

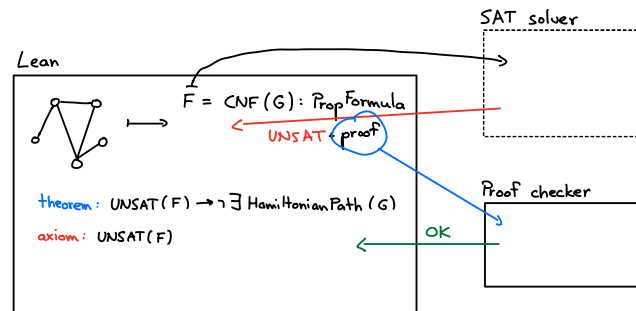
Hamiltonian Paths

NP-hard, in general no easy certificate for non-existence.
Use a SAT solver.



Hamiltonian Paths

NP-hard, in general no easy certificate for non-existence.
Use a SAT solver.



Take-aways for incorporating a database into a proof assistant

It probably won't be what you expected

- It depends on the database.
- Lean may be a sensible proof assistant to start with. If you do choose Lean, a lot depends on Mathlib.
- Checking a database may force you to consider efficiency and may make you feel like you are doing CS 50 years ago.
- Alternative to our approach: formal verification of algorithms.
- We implore database designers to consider certificates whenever possible.

We found it particularly advantageous to minimize the amount of computation performed directly by Lean, especially in situations involving meta-programming, where Lean metaprograms construct proofs for each value.

It would be possible to implement most of the properties of graphs in HoG. In some cases, however, we did not see a clear way out. For instance, computing the maximum or minimum eigenvalues of the adjacency matrix would require not only a standard format for algebraic numbers and a trusted, efficient computation engine for them, but also further considerations if we wanted to reason about extremality.

Thank you!

Recap of what we have been up to

- **MathBases**

*Adam Towsley, Ben Spitz,
David Roe, David Lowry-Duda,
Benjamin Hutz, Edgar Costa,
KB*

- **1000+ theorems**

*Freek Wiedijk, Floris van Doorn,
KB; editors for each system*

- **Lean-HoG**

*Jure Taslak, Gauvain Devillez,
KB, Andrej Bauer*



Slides (hopefully) and references at katja.nat.si, as promised



Conference on Intelligent
Computer Mathematics

CICM 2026 @Ljubljana

September 21-25

The website is online