

Type theory for discrete mathematics

Andrej Bauer

University of Ljubljana

*19th Conference on Intelligent Computer Mathematics (CICM)
Ljubljana, Slovenia, 21–25 September 2026*

Talk outline

- ▶ **Enumix**

- ▶ a domain-specific type theory for discrete mathematics
- ▶ joint work with Jure Taslak

- ▶ **MathQL**

- ▶ a query language for mathematical databases
- ▶ joint work with Danel Ahman and Lucy Horowitz

Part I

Enumix

Overview of Enumix

- ▶ *EnumixTT*
 - ▶ domain-specific type theory for discrete mathematics
 - ▶ semantic model: finite groupoids
 - ▶ embedded in Lean

Overview of Enumix

- ▶ *EnumixTT*
 - ▶ domain-specific type theory for discrete mathematics
 - ▶ semantic model: finite groupoids
 - ▶ embedded in Lean
- ▶ *Enumix*
 - ▶ domain-specific language for enumeration
 - ▶ EnumixTT compiles to Enumix
 - ▶ (slow) reference implementation in Lean
 - ▶ future: optimizing compiler

The category of finite groupoids $\mathbf{FinGpd}$

- ▶ A groupoid is a category with invertible morphisms.
- ▶ $\mathbf{FinGpd}$: finite groupoids and functors.

The category of finite groupoids $\mathbf{FinGpd}$

- ▶ A groupoid is a category with invertible morphisms.
- ▶ $\mathbf{FinGpd}$: finite groupoids and functors.
- ▶ Examples:
 - ▶ a group is a single-object groupoid,
 - ▶ a set is a *discrete* groupoid.

The category of finite groupoids $\mathbf{FinGpd}$

- ▶ A groupoid is a category with invertible morphisms.
- ▶ $\mathbf{FinGpd}$: finite groupoids and functors.
- ▶ Examples:
 - ▶ a group is a single-object groupoid,
 - ▶ a set is a *discrete* groupoid.
- ▶ Finite sets and bijections form a subcategory $\mathbf{FinSet}$.

EnumixTT

Types:

► $A \times B, A + B, A \rightarrow B, \Sigma_{(x:A)} B(x), \Pi_{(x:S)} B(x)$

EnumixTT

Types:

- ▶ $A \times B, A + B, A \rightarrow B, \Sigma_{(x:A)} B(x), \Pi_{(x:S)} B(x)$
- ▶ subset type $\{x:A \mid \phi(x)\}$

EnumixTT

Types:

- ▶ $A \times B, A + B, A \rightarrow B, \Sigma_{(x:A)} B(x), \Pi_{(x:S)} B(x)$
- ▶ subset type $\{x:A \mid \phi(x)\}$
- ▶ identity type $\text{Id}_A(s, t)$ is isomorphism $s \cong_A t$

EnumixTT

Types:

- ▶ $A \times B, A + B, A \rightarrow B, \Sigma_{(x:A)} B(x), \Pi_{(x:S)} B(x)$
- ▶ subset type $\{x:A \mid \phi(x)\}$
- ▶ identity type $\text{Id}_A(s, t)$ is isomorphism $s \cong_A t$
- ▶ finite sets $[n] := \{0, 1, \dots, n-1\}$

EnumixTT

Types:

- ▶ $A \times B, A + B, A \rightarrow B, \Sigma_{(x:A)} B(x), \Pi_{(x:S)} B(x)$
- ▶ subset type $\{x:A \mid \phi(x)\}$
- ▶ identity type $\text{Id}_A(s, t)$ is isomorphism $s \cong_A t$
- ▶ finite sets $[n] := \{0, 1, \dots, n-1\}$
- ▶ the symmetric group S_n qua groupoid BS_n :
 - ▶ object $\star$, morphisms are permutations on $[n]$

Logic:

- ▶ $\perp, \top, \vee, \wedge, \neg, \forall, \exists, =$

EnumixTT

Types:

- ▶ $A \times B, A + B, A \rightarrow B, \Sigma_{(x:A)} B(x), \Pi_{(x:S)} B(x)$
- ▶ subset type $\{x:A \mid \phi(x)\}$
- ▶ identity type $\text{Id}_A(s, t)$ is isomorphism $s \cong_A t$
- ▶ finite sets $[n] := \{0, 1, \dots, n-1\}$
- ▶ the symmetric group S_n qua groupoid BS_n :
 - ▶ object $\star$, morphisms are permutations on $[n]$

Logic:

- ▶ $\perp, \top, \vee, \wedge, \neg, \forall, \exists, =$
- ▶ $s = t$ is interpreted as $\exists \alpha : s \cong_A t. \top$

EnumixTT

Types:

- ▶ $A \times B, A + B, A \rightarrow B, \Sigma_{(x:A)} B(x), \Pi_{(x:S)} B(x)$
- ▶ subset type $\{x:A \mid \phi(x)\}$
- ▶ identity type $\text{Id}_A(s, t)$ is isomorphism $s \cong_A t$
- ▶ finite sets $[n] := \{0, 1, \dots, n-1\}$
- ▶ the symmetric group S_n qua groupoid BS_n :
 - ▶ object $\star$, morphisms are permutations on $[n]$

Logic:

- ▶ $\perp, \top, \vee, \wedge, \neg, \forall, \exists, =$
- ▶ $s = t$ is interpreted as $\exists \alpha : s \cong_A t. \top$
- ▶ decidable: all quantifiers are finite

EnumixTT

Types:

- ▶ $A \times B, A + B, A \rightarrow B, \Sigma_{(x:A)} B(x), \Pi_{(x:S)} B(x)$
- ▶ subset type $\{x:A \mid \phi(x)\}$
- ▶ identity type $\text{Id}_A(s, t)$ is isomorphism $s \cong_A t$
- ▶ finite sets $[n] := \{0, 1, \dots, n-1\}$
- ▶ the symmetric group S_n qua groupoid BS_n :
 - ▶ object $\star$, morphisms are permutations on $[n]$

Logic:

- ▶ $\perp, \top, \vee, \wedge, \neg, \forall, \exists, =$
- ▶ $s = t$ is interpreted as $\exists \alpha : s \cong_A t. \top$
- ▶ decidable: all quantifiers are finite
- ▶ isomorphism invariance: if $\phi(s)$ and $s \cong t$ then $\phi(t)$

Example: labeled vs. unlabeled magmas

- ▶ Labeled magma on n elements:

$$[n] \times [n] \rightarrow [n]$$

Example: labeled vs. unlabeled magmas

- ▶ Labeled magma on n elements:

$$[n] \times [n] \rightarrow [n]$$

- ▶ Define $\text{El}_n : \text{BS}_n \rightarrow \text{FinSet}$:

$$\text{El}_n(\star) := [n]$$

$$\text{El}_n(\pi) := \pi$$

Example: labeled vs. unlabeled magmas

- ▶ Labeled magma on n elements:

$$[n] \times [n] \rightarrow [n]$$

- ▶ Define $\text{El}_n : \text{BS}_n \rightarrow \text{FinSet}$:

$$\text{El}_n(\star) := [n]$$

$$\text{El}_n(\pi) := \pi$$

- ▶ Unlabeled magma on n elements:

$$\Sigma_{(x : \text{BS}_n)} (\text{El}(x) \times \text{El}(x) \rightarrow \text{El}(x))$$

Witnessing finiteness of a set S

Any of the following:

1. given $n : \mathbb{N}$ and $f : \mathbb{N} \cong S$
2. proof of $\exists n : \mathbb{N}. \exists f : \mathbb{N} \cong S. \top$
3. an Enumix program that emits elements of S

Witnessing finiteness of a set S

Any of the following:

1. given $n : \mathbb{N}$ and $f : \mathbb{N} \cong S$
2. proof of $\exists n : \mathbb{N}. \exists f : \mathbb{N} \cong S. \top$
3. an Enumix program that emits elements of S

Enumix:

- ▶ small untyped DSL for enumerating finite sets
- ▶ semantics: finite powerset monad
- ▶ reference implementation in Lean, formally verified

Witnessing finiteness of a set S

Any of the following:

1. given $n : \mathbb{N}$ and $f : \mathbb{N} \cong S$
2. proof of $\exists n : \mathbb{N}. \exists f : \mathbb{N} \cong S. \top$
3. an Enumix program that emits elements of S

Enumix:

- ▶ small untyped DSL for enumerating finite sets
- ▶ semantics: finite powerset monad
- ▶ reference implementation in Lean, formally verified

Not discussed here:

- ▶ witness finiteness of families and functors also
- ▶ implementation in Lean on top of Mathlib

How to enumerate a groupoid

Enumerate objects of a groupoid A *up to isomorphism*:

- ▶ enumerate objects of A , and
- ▶ filter out those isomorphic to already seen ones.

How to enumerate a groupoid

Enumerate objects of a groupoid A *up to isomorphism*:

- ▶ enumerate objects of A , and
- ▶ filter out those isomorphic to already seen ones.

We use *footprints* to speed up isomorphism testing:

- ▶ a footprint on A is a functor $f : A \rightarrow S$ to a set S ,
- ▶ if $f(x) \neq f(y)$ then $x \not\cong_A y$, can be a cheap test
- ▶ any definable $f : A \rightarrow S$ is a footprint,
- ▶ user can specify footprints, otherwise defaults are used.

Observations

- ▶ Tight connection between *math* and *computation*.
- ▶ Formalization is the norm.
- ▶ We want an efficient Enumix interpreter.
- ▶ Slower than specialized enumerators, but *general*.
- ▶ Other uses: specification of mathematical datasets.

Part II

MathQL

Joint work Danel Ahman and Lucy Horowitz.

A query language for mathematics

A query language for mathematics

- ▶ *Safe*: the database cannot be corrupted

A query language for mathematics

- ▶ *Safe*: the database cannot be corrupted
- ▶ *Expressive*: complex queries are supported

A query language for mathematics

- ▶ *Safe*: the database cannot be corrupted
- ▶ *Expressive*: complex queries are supported
- ▶ *Abstract*: *mathematical* queries, not database queries

A query language for mathematics

- ▶ *Safe*: the database cannot be corrupted
- ▶ *Expressive*: complex queries are supported
- ▶ *Abstract*: *mathematical* queries, not database queries
- ▶ *Meaningful*: senseless queries are refused

Example

*“Chromatic number of cubic graphs with
abelian automorphism group.”*

Example

*“Chromatic number of cubic graphs with
abelian automorphism group.”*

$\{(\text{id}(g), g.\text{chromatic_number}) \mid g \in \text{Graph} \mid$
 $g.\text{is_regular} \wedge g.\text{max_degree} = 3 \wedge$
 $g.\text{automorphism_group}.\text{is_abelian}\}$

Example

“Chromatic number of cubic graphs with abelian automorphism group.”

$$\{(\text{id}(g), g.\text{chromatic_number}) \mid g \in \text{Graph} \mid \\ g.\text{is_regular} \wedge g.\text{max_degree} = 3 \wedge \\ g.\text{automorphism_group}.\text{is_abelian}\}$$

```
SELECT c1."graph6" AS c2, c1."chromatic_number" AS c3
FROM "graph" AS c1
LEFT JOIN "small_group" AS c4
  ON c4."order" = c1."aut_group_order"
  AND c4."index" = c1."aut_group_index"
WHERE ((c1."is_regular" AND (c1."max_degree" = 3))
      AND c4."is_abelian")
```

MathQL

- ▶ Typed domain-specific language
- ▶ It attaches to an SQL database
- ▶ It compiles MathQL to SQL
- ▶ Implemented in Lean as a certifying compiler
- ▶ Implemented in Python as MCP

Bridge MCP

- ▶ Small database enclosed:
 - ▶ groups up to order 127 with 19 invariants
 - ▶ graphs up to 8 vertices with 20 invariants
 - ▶ π -base database of topological spaces and properties
 - ▶ provenance information for each column
- ▶ AI agent can query the database in MathQL

Demo